Knowledge Management Techniques to Augment Textual CBR (PhD Thesis)

Padmanabhan, D. Knowledge Management Techniques to Augment Textual CBR (PhD Thesis)

KNOWLEDGE MANAGEMENT TECHNIQUES TO AUGMENT TEXTUAL CBR

A THESIS

submitted by

DEEPAK S PADMANABHAN

for the award of the degree

of

DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

March 2014

THESIS CERTIFICATE

This is to certify that the thesis titled **KNOWLEDGE MANAGEMENT TECHNIQUES TO AUGMENT TEXTUAL CBR**, submitted by **Deepak S Padmanabhan (CS09D010)**, to the Indian Institute of Technology, Madras, for the award of the degree of **Doctor of Philosophy**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Deepak Khemani
Research Guide
Professor
Dept. of Comp. Sc. & Engg.,
IIT Madras, Chennai

Dr. Sutanu Chakraborti
Research Guide
Assistant Professor
Dept. of Comp. Sc. & Engg.,
IIT Madras, Chennai

Place: Chennai

Date: 17th March, 2014

ACKNOWLEDGEMENTS

I am deeply thankful to my supervisors, Dr. Deepak Khemani and Dr. Sutanu Chakraborti, without whose encouragement and support, this thesis would not have been possible. Whenever I visited IIT Madras in search for ideas to decide on the next step in terms of technical directions, they were always ready to spend many hours, and sometimes even entire afternoons, discussing fine technical details. My initial lack of familiarity with the field of Textual Case-based Reasoning was well offset by the points that I have picked up during such meetings. My heartfelt thanks are also due to Dr. Raghu Krishnapuram for being my external advisor.

I am grateful to Dr. Karthik Visweswariah for technical inputs and feedback that were very useful to help getting the work published in good venues. I would also like to thank Dr. Nirmalie Wiratunga and her students, especially Ibrahim Adeyanju and Sadiq Sani, for insightful discussions during my visits to the Robert Gordon University, Aberdeen. I also would like to express my gratitude to Dr. Balaraman Ravindran who was always enthusiastic in providing constructive feedback during DC meetings and informal discussions. I also extend my thanks to the staff at the Department of CSE who provided all necessary support.

ABSTRACT

KEYWORDS: Case-based Reasoning; Textual CBR; Knowledge Management; Text Mining

The past decade had witnessed an unprecedented growth in the amount of available digital content, and its volume is expected to continue to grow the next few years. Unstructured text data generated from web and enterprise sources form a large fraction of such content. Many of these contain large volumes of reusable data such as solutions to frequently occurring problems, and general know-how that may be reused in appropriate contexts. In this work, we address issues around leveraging unstructured text data from sources as diverse as the web and the enterprise within the Case-based Reasoning framework. Case-based Reasoning (CBR) provides a framework and methodology for systematic reuse of historical knowledge that is available in the form of problem-solution pairs, in solving new problems.

Here, we consider possibilities of enhancing Textual CBR systems under three main themes: *procurement*, *maintenance* and *retrieval*. We adapt and build upon the state-of-the-art techniques from data mining and natural language processing in addressing various challenges therein. Under *procurement*, we investigate the problem of extracting cases (i.e., problem-solution pairs) from data sources such as incident/experience reports. We develop case-base *maintenance* methods specifically tuned to text targeted towards retaining solutions such that the utility of the filtered case base in solving new problems is maximized. Further, we address the problem of query suggestions for textual case-bases and show that exploiting the problem-solution partition can enhance *retrieval* effectiveness by prioritizing more useful query suggestions. Additionally, we illustrate interpretable clustering as a tool to drill-down to domain specific text collections (since CBR systems are usually very domain specific) and develop techniques for improved similarity assessment in social media sources such as microblogs. Through extensive empirical evaluations, we illustrate the improvements that we are able to achieve over the state-of-the-art methods for the respective tasks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 The Data Deluge

We are witnessing an explosive growth in the amount of digitized data. More people use computers than ever before, and tremendous amounts of data are being created every minute. The amount of data available today has crossed exabyte (10^{18} bytes) levels and is now in the zettabyte (10^{21} bytes) range. Recent studies[1,2] assess that the amount of digitized data would grow from 1.8 zettabytes in 2011 to 7.9 zettabytes in 2015 as illustrated in Figure 1.1. To put this in perspective, it has been estimated that all of human speech ever spoken by mankind only adds up to 42 zettabytes[3].

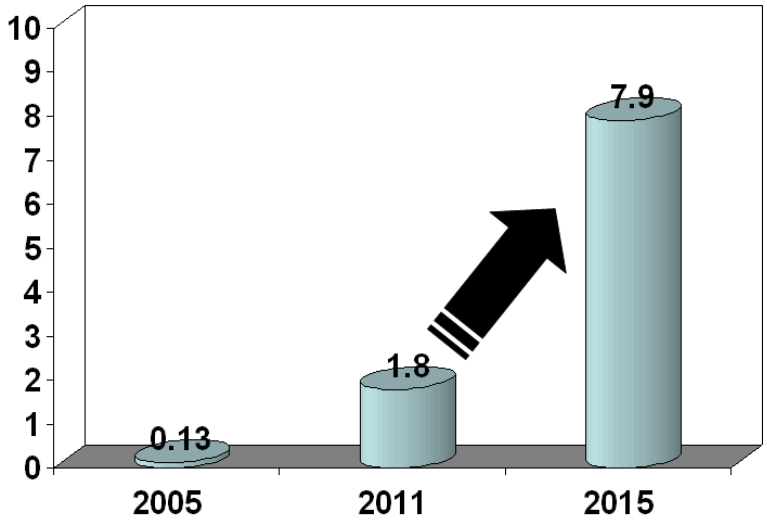

Figure 1.1: Data Growth in Zettabytes

The explosive growth of data is not necessarily limited to the realm of traditional kinds of data that are in structured form such as bank transactions and digitized entity data from new domains. A significant fraction of the data being generated every day

[1]http://www.dataversity.net/the-growth-of-unstructured-data-what-are-we-going-to-do-with-all-those-zettabytes/ accessed February 5^{th}, 2014

[2]http://www.datacenterknowledge.com/archives/2011/06/28/digital-universe-to-add-1-8-zettabytes-in-2011/ accessed February 5^{th}, 2014

[3]http://itre.cis.upenn.edu/myl/languagelog/archives/000087.html accessed February 5^{th}, 2014

are in the form of unstructured text. Data generated in the web in text form include blogposts, web pages and social media content. Email, probably the most popular form of communication in the digital era, also mostly comprises text data. It was estimated that 107 trillion emails were sent in the year 2010[4]; conservatively assuming that an average email contains 15kb of data, the amount of data generated through email on a daily basis stands at 4k terabytes. The estimated amount of text data generated daily from various sources such as Facebook[5], blogs[6] and email (as estimated above) is shown in Figure 1.2. Though statistics are not available on what proportion of all data is unstructured, it has been estimated that the fraction of unstructured data in the enterprise context is a staggering 80%[7].

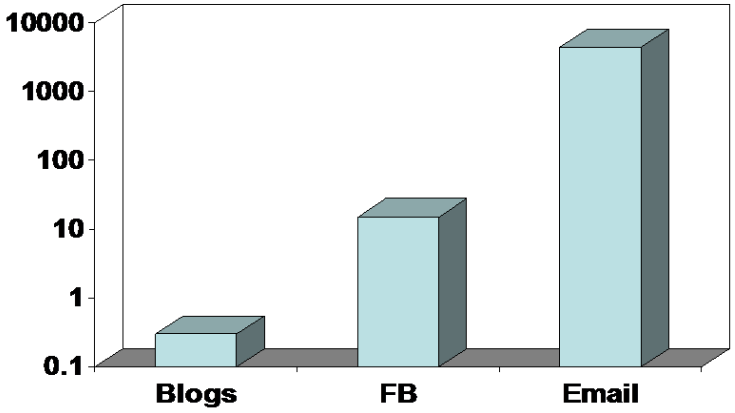

Figure 1.2: Text content generated daily (in Terabytes)

1.2 Text Mining

This explosion of text data has led to an increased focus on techniques for text data analytics. Text Analytics encompasses a wide array of techniques to deal with text data, some of which are as follows:

- **Text Clustering:** Clustering is the task of grouping datasets into clusters of coherent data objects. Text clustering techniques (Steinbach *et al.*, 2000) operate on sets of text documents and group them into clusters of *similar* text documents;

[4]http://phys.org/news/2011-01-trillion-emails-year-pingdom.html accessed February 5^{th}, 2014
[5]http://www.cs.kent.edu/ jin/Cloud12Spring/BigData.pptxâĂŐ accessed February 4^{th}, 2014
[6]http://mashable.com/2012/06/22/data-created-every-minute/ accessed February 4^{th}, 2014
[7]http://www.eweek.com/c/a/Data-Storage/Managing-Unstructured-Data-in-the-Cloud-12-Factors-to-Consider-215018/ accessed February 5^{th}, 2014

similarity is assessed by means of measures that quantify the number of common words between text documents.

- **Text Classification:** When provided with text documents with labels as to which category (aka *class*) they belong, text classifiers (Aas and Eikvil, 1999) build statistical models that learn characteristics of each class. Once such models are built, they can be used to estimate the membership of a new text document to a particular class.

- **Text Segmentation:** Since most text documents deal with multiple topics, text documents often need to be segmented into coherent segments. Text segmentation algorithms (Reynar, 1998) identify segment boundaries using various statistical features such as the difference in lexical character between text segments on either side of the candidate segment boundaries.

- **Text Retrieval:** Text retrieval, or Information Retrieval (Baeza-Yates and Ribeiro-Neto, 1999), as it is commonly called, is the task of finding relevant text documents in response to a search query that typically comprises a few words. The classical text retrieval engine works on corpora of raw text documents, whereas modern retrieval engines such as web search engines operate on more complex text data that include links and semi-structured content.

A bag-of-words representation of text documents, and a similarity measure between such representations are employed by most of the above techniques. We will get into details of standard similarity measures in Chapter 2. All the above techniques are directed towards making text data easier to use by techniques such as *labeling* (as in classification or segmentation), *grouping* (e.g., clustering) and *retrieval*. Rapid advances in such text mining techniques over the decades have led to enabling the usage of a text data to solve a large number of real-world problems, the most common example being web search.

1.3 Case-based Reasoning and Textual CBR

We now turn our attention to an Artificial Intelligence-based problem solving paradigm called *Case-based Reasoning (CBR)* (Kolodner, 1992). Though we will defer a detailed explanation of CBR to Chapter 2, we provide a brief summary herein. Case-based Reasoning uses two major assumptions, viz., (1) Similar problems have similar solutions, and (2) Problems recur, though not necessarily identically. Case-based Reasoning operates over a large corpus of data called the *case base*, which is comprised of problem-solution pairs (called *cases*). When a user poses a problem, similar problems are *retrieved* from the case base, and an attempt is made to *reuse* their solutions for the new problem. However, most often, the solutions from the case base need to be *revised* or adapted to solve the new problem, following which the problem-solution pair is *retained* in the case base. Research in CBR is aimed at improving the four *Rs*, the *retrieval, reuse, revise* and *retain* processes, to help improve CBR-based problem solving. Case-based Reasoning has traditionally focused on case bases that are structured; a case base for a trip planning system could include a problem section that is *Source: X, Destination: Y* and a solution *{ Part1: [Type:Walk, Source:X, Destination:A], Part 2: [Type:Bus, Source:A, Destination:B], Part3: [Type:Car, Source:B, Destination:Y] }* that describes a 3-part plan to travel from X to Y.

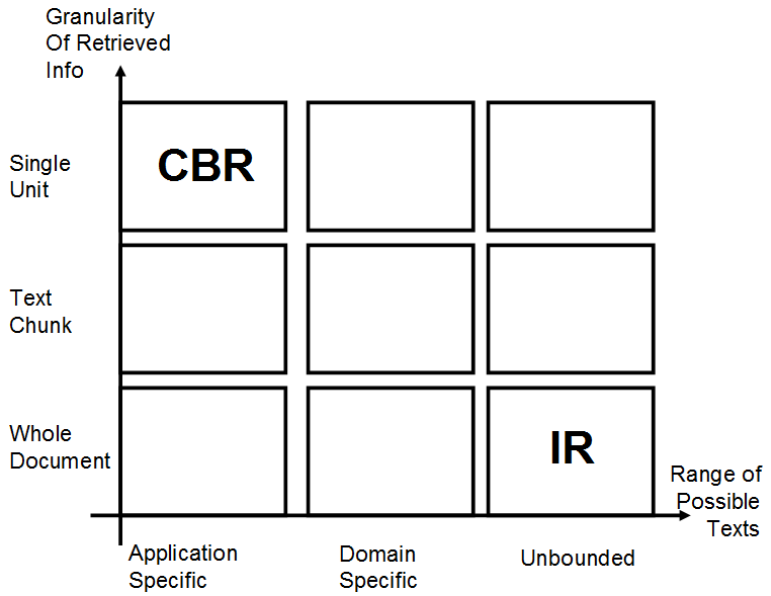

Figure 1.3: Textual CBR and IR (Adapted from Brown *et al.* (1998))

Textual CBR (Lenz *et al.*, 1998), a more recent sub-field of CBR, looks at CBR in cases where the cases are partially or fully expressed in text. With Textual CBR,

CBR stepped into an area traditionally addressed by Information Retrieval techniques. However, Textual CBR may be differentiated from traditional IR techniques in many aspects (graphic in Figure 1.3):

- **Range of Possible Texts:** Information Retrieval is largely a domain independent technique, focusing on generic similarity measures between text documents. Thus, the range of possible domains that IR systems can address is bounded only by the availability of text data. On the other hand, Textual CBR is usually focused and tuned to a specific application domain. In classical CBR that uses unstructured text, the similarity measures may be extremely domain specific (e.g., techniques for comparing travel plans is typically not applicable for comparing product descriptions); at the least, text similarity measures may be adapted by weighing words according to a domain ontology while comparing text segments. Integrating other kinds of domain knowledge is also considered to be critical to Textual CBR.

- **Granularity of Solution:** Information Retrieval focuses on ranking documents in response to a user query whereas CBR looks to provide specific information that is relevant to solving the new problem posed by the user. Thus, IR may be used to address cases not necessarily related to problem solving; a typical case is that of an *exploratory user* trying to find more information about the concepts described in the keywords in the query. Thus, IR methods provide the user with top-ranked documents whereas solutions provided by Textual CBR systems could be more finer in granularity.

A graphical comparison on the two criteria above appears in Figure 1.3. A more comprehensive comparison of Textual CBR and IR appears in Lenz (1998). Textual CBR may be contrasted with other fields such as Text Mining, Natural Language Generation and Text Summarization too; (Adeyanju, 2011) provides a detailed comparison. With recent advances in many of these fields, it needs to be said that the differences are getting blurred. Thus, many of the core processes of Textual CBR are fairly specialized in their own right, and knowledge management techniques from other communities (such as above) may not be readily usable. For example, harnessing *domain knowledge*

to *extract* to-the-point solutions from solutions of *similar problems* is quite a challenging problem, and the combination of domain knowledge usage and text similarity is fairly unique to Textual CBR. Text Reuse has been been recently addressed extensively in (Adeyanju, 2011). The focus of this thesis is to develop knowledge management techniques to *augment* and *enrich* Textual CBR, as we will describe in the next section. In particular, the work described is to develop knowledge management techniques for problems inspired by use cases towards improving Textual CBR.

1.4 Research Motivation and Problems Addressed

The work presented in this thesis describe techniques to enhance Textual CBR by developing techniques that could be used to augment the traditional CBR processes. We focus on three main themes, *procurement*, *maintenance* and *retrieval*. We now briefly summarize the work presented under these heads:

- **Procurement:** Here, we mainly focus on developing techniques to help rapidly procure large case bases from unstructured text data. CBR systems are only as good as their case bases (since that is the primary resource for problem solving), and thus, enriching case bases by harvesting problem-solution pairs from unstructured text data is important to ensure widespread applicability of CBR systems in an era of massive creation of digitized knowledge.We describe techniques for extracting case bases from the vast amounts of textual knowledge in the public web and enterprises.

- **Maintenance:** Though large case bases provide more knowledge to work with, large case bases are hard to manage and also slow down retrieval. Case base maintenance focuses on techniques for weeding out cases in a controlled manner ensuring that the effectiveness of the CBR system is minimally affected. We describe how advances in text mining could be leveraged to define a statistical notion of usability of solutions to problems in the absence of supervised information such as pairwise solution similarities. We then outline a technique to use such usability estimates to effectively compact case bases in tune with pre-specified

preferences.

- **Retrieval:** With the popularity of web search that works with user intent specified with just 2-3 keywords, adoption of Textual CBR systems would be accelerated by a user interface that could work with such scanty information as well. However, traditional CBR systems expect a reasonably well-specified (even if it be a partial specification) problem description; we focus on query suggestion as a support mechanism to help the user author better problem specifications, and develop a query suggestion technique tuned to retrieval over textual case bases. Another realm where scanty text is the norm is that of microblogs. We look into the problem of developing improved similarity measures for microblog text as a first step towards bringing them under consideration for CBR, similarity assessment being a fundamental CBR functionality.

The work presented in this thesis falls under the category of techniques that are often referred to as *knowledge discovery* methods. Knowledge Discovery is usually meant to encompass efforts ranging from organizing contributed knowledge (e.g., Freebase[8]) to learning deep semantic relationships between entities (e.g., Yago[9] and NELL[10]). In this thesis, we address specific problems (as outlined above) in the realm of extracting problem-solution data and processing problems-solution data; while generic knowledge discovery techniques may be adapted towards solving the problems we address, specialized techniques are often necessary to achieve desirable accuracy levels for problem-solution data processing as we will show through empirical analyses. We will describe the three themes in more detail, with illustrations as to how the techniques we propose would fit in an extended CBR workflow, in separate sections herein.

1.4.1 Procurement of Case bases from Text Data

Under this theme, we look into various challenges in exploiting the explosive growth of unstructured text data, to enrich Textual Case-based Reasoning systems. A very simple way to augment a CBR system is to add to the case base that it works with. We outline a

[8]http://www.freebase.com accessed February 5^{th}, 2014
[9]http://www.mpi-inf.mpg.de/yago-naga/yago/ accessed February 5^{th}, 2014
[10]http://rtw.ml.cmu.edu/rtw/ accessed February 5^{th}, 2014

three stage process for such augmentation of case bases in Figure 1.4. We will describe the three phases briefly below.

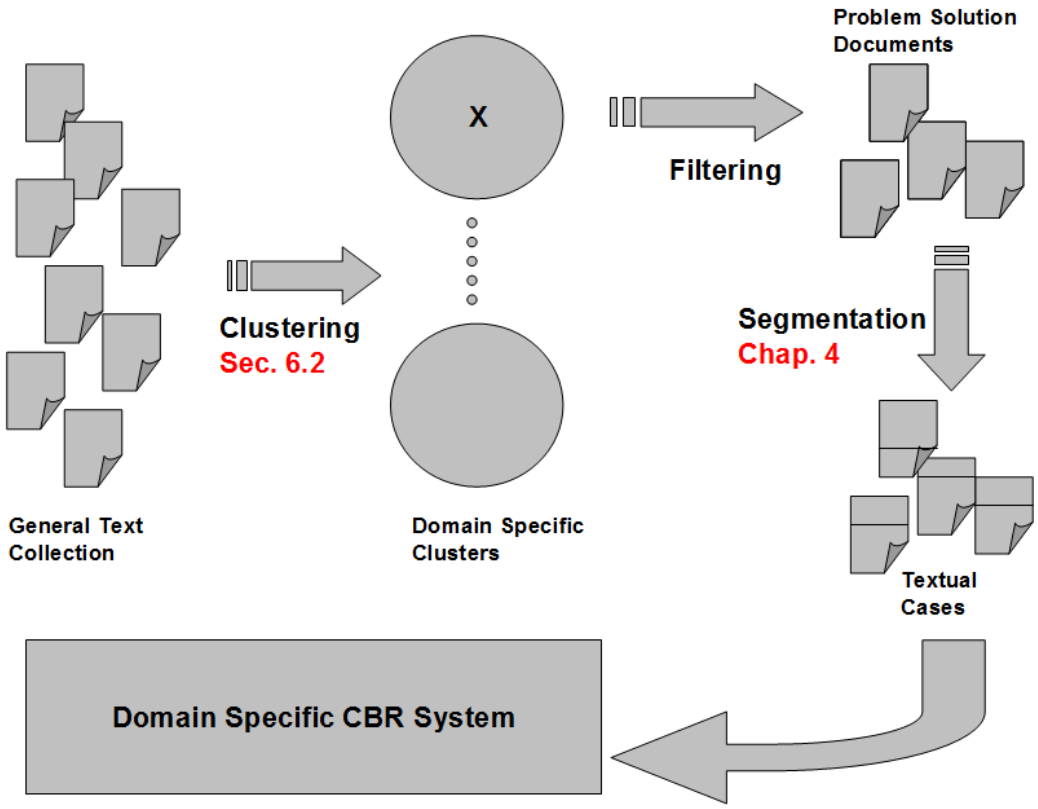

Figure 1.4: Procurement Phase

Clustering: As outlined in Figure 1.3, most CBR systems are specialized to a particular domain. Thus, supplementing a specific Textual CBR system obviously involves finding a subset of text documents that are of relevance to the domain that the system deals with. We consider clustering as a method to scope down the large generic text datasets into a subset specialized to the domain of interest. Though clustering is a very well-studied topic, we argue that rapidly narrowing down on documents pertaining to the domain of interest would be facilitated by improved cluster descriptions so that a user can sift through such cluster descriptions and make cluster elimination decisions easily. Interpretable clustering has been proposed as a means of expediting human understanding of content within clusters. In Section 6.2, we propose an interpretable clustering method that improves upon the state-of-the-art by reducing cluster description sizes by close to an order of magnitude (since smaller descriptions are often preferred (Gao and Ester, 2006)), while retaining fairly good accuracy in clustering. As illustrated in Figure 1.4, clustering is part of a pre-filtering process that would group a general text collection into domain specific clusters.

Filtering: From among the domain specific collection that matches the domain of interest of the CBR system, there could only be a few documents that have a problem-solution format. Infact, we do not need the document to adhere to a problem-solution format if we have the technology to extract problem and solution fragments from the text of the document; however, given the state-of-the-art, such extraction would be possible only if there is enough labeled data. In the absence of plentiful labeled data, we could just restrict ourselves to identifying documents of the type that have a problem-solution format. Such formats are commonly seen in *report*-like data such as incident reports, bug reports and medical diagnosis documents. The filtering stage refers to identifying such documents from the domain specific cluster discovered from the clustering process. This problem is quite hard and requires domain expertise to be effectively tackled; hence, we do not consider automatic methods for solving the filtering problem in this work. Manually identifying documents that contain problem-solution information from a cluster could however be aided by any available keyword search tool; for example, in the medical domain where one may want to identify diagnosis reports (where the symptoms are analogous to the problem, and the actual diagnosis is akin to the solution), a script that takes a list of disease names from an ontology and looks for only those documents that contain at least one disease name towards the end could be of use to filter out a large fraction of documents. Text style-based classification techniques (e.g., (Stamatatos *et al.*, 2000)) may be suitably adapted to filter the results of such keyword search to drill-down to documents containing problem and solution information. Another heuristic to identify solution documents is to build classifiers that select documents containing a sequence of steps written in imperative style, since detailed solutions often assume such forms. In very specialized domains where such ontologies may not be available, laborious manual inspection to identify any problem-solution information may be the only way out.

Segmentation: Documents such as incident and bug reports are typically written post facto and mostly for audit purposes. Though they contain problem and solution parts and are structured in a two-part fashion, these are not cleanly partitioned into segments that talk about the problems and solutions within them. Since we would like to use them as cases, we would need a problem-solution partition, motivated by the two-part structure of cases. In most domains, such documents are available in collections of similar reports, and thus, there is a possibility of effectively making use of collec-

tion level knowledge to segment individual reports. We investigate such possibilities and develop an approach (that we will describe in Chapter 4) that uses langauge and translation models to segment such text documents into the component problem and solution parts. Once such segmentation is done, it is straightforward to use such textual problem-solution tuples to augment existing case bases.

1.4.2 Maintenance of Textual Case Bases

Case base maintenance deals with the problem of filtering cases by removing those with low utility towards answering new problems. With injection of a large amount of cases into textual case bases by using processes such as those listed above, the maintenance problem becomes even more critical. It is often the case that specific to-the-point solutions are highly valued for niche problems, whereas more generic solutions that can solve a wider variety of problems are valued for their generality. Since the desired generality of solutions is often scenario-dependent, we propose a two-phase textual case base maintenance approach that can cater to user-specified trade-offs in this regard.

Usability Estimation: A method to quantify usability of a solution in the case base to any problem in the case base would provide a powerful primitive that can be used in the compaction process. We often need to deal with problems that have multiple associated solutions, such as is common in the case of datasets derived from community-driven question answering systems. Most such systems have multiple solutions per problem and votes with each solution signifying their popularities; this provides ample redundancy to effectively estimate the usability of a solution to a problem by usage of text similarity measures from the information retrieval community. We describe a usability estimation technique for textual case bases in Section 3.6.

User-tuned Case base Compaction: With the availability of a usability computation method, the usability of any solution across all problems in the case base may be easily obtained. Generic solutions would be usable across many problems, whereas specific solutions are likely to be highly usable for a small number of problems. Retaining generic solutions would enable the CBR system to provide many (generic) solutions for user-posed problems whereas retaining specific solutions has its own advantages too. We outline a case-base compaction technique that can heed to a user-specified

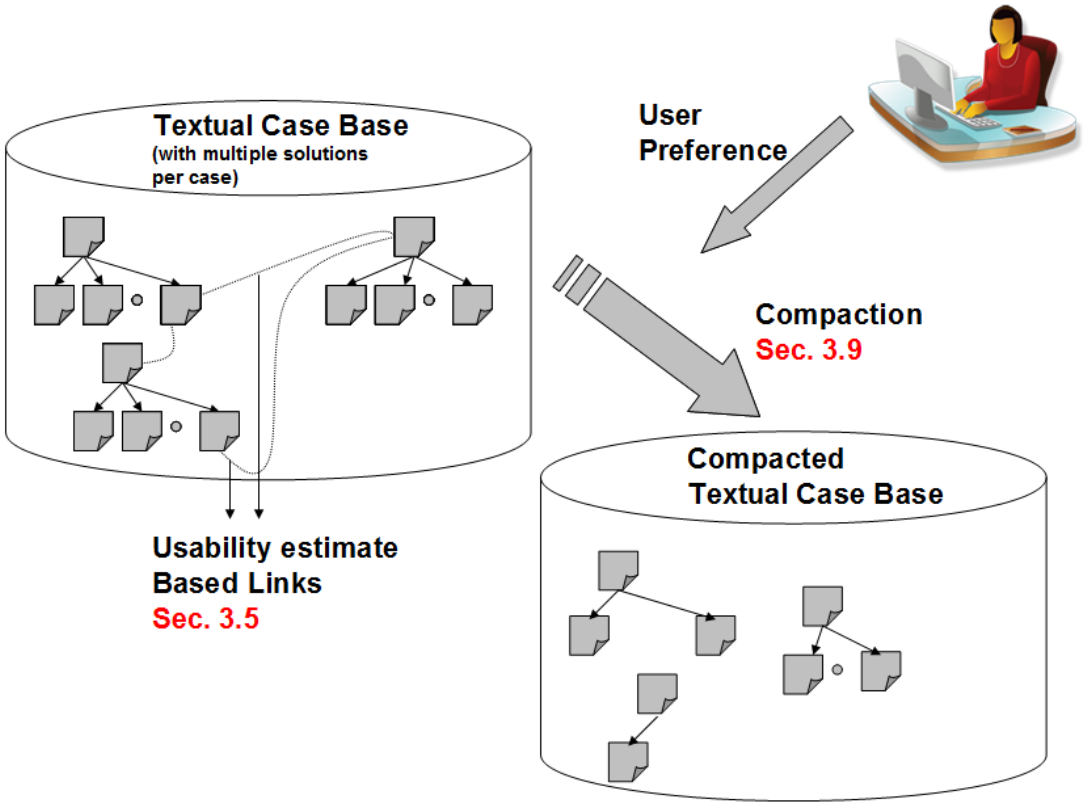

Figure 1.5: Maintenance Phase

trade-off on various criteria such as generality of solutions and number of solutions in Section 3.9. Through an extensive set of experiments, we demonstrate the efficacy of the proposed technique on various evaluation measures.

The schematic in Figure 1.5 illustrates the operation of the two-phases.

1.4.3 Retrieval

We address two specific challenges that pertain to retrieval in textual case bases; one relates to providing support to the user for authoring queries to a Textual CBR system, and another that pertains to similarity assesment for social media data that need some special treatment due to intrinsic noise. Figure 1.6 is a pictorial overview of the techniques that we propose.

Query Suggestions: The primary method of interacting with text retrieval systems such as search engines has been through queries that are short phrases consisting of 2-3 words, on an average. Though CBR systems expect a more detailed problem description, Textual CBR systems would most likely have to deal with user queries that are

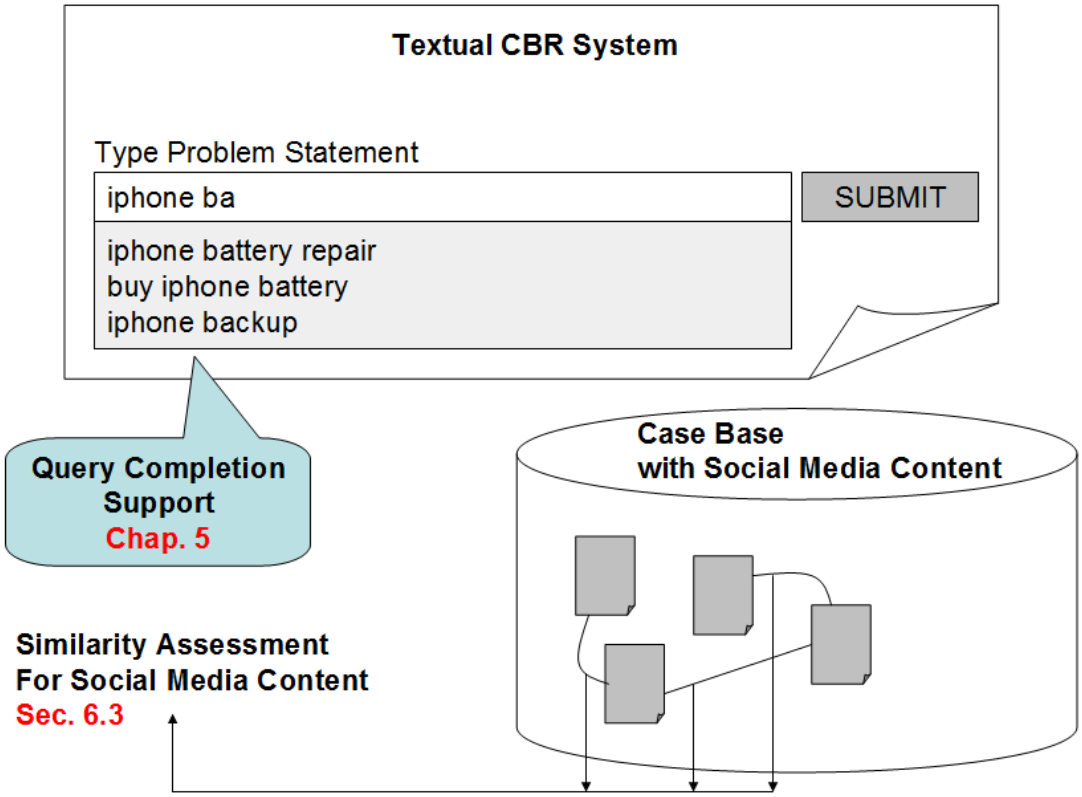

Figure 1.6: Challenges in Retrieval

more search-like, i.e., comprising of a small set of words, or a phrase. Search engines have used query suggestions as a feature to help the user enter queries, and query suggestions that are derived from the underlying corpora could help guide the user in using words that are abundant in the corpora (rather than substitute words such as synonyms etc.), in turn helping more accurate similarity assessments and consequent better retrieval. We address the query suggestion problem in the context of textual case bases in Chapter 5, and illustrate that using case-base alignment measures in prioritizing phrases as query suggestions can significantly improve query suggestion accuracy.

Microblog Similarity Assessment: The CBR retrieval process usually employs domain specific similarity measures. On the other hand, with unstructured text data, domain-independent text similarity measures are an obvious first choice due to their simplicity and lack of reliance on domain expertise. With the advent of short and noisy text segments from sources such as microblogs, text similarity measures that rely on quantifying the similarity based on the number of common words may not fare well. We discuss an improved technique for similarity assesment between microblog texts in Section 6.3, and empirically establish the effectiveness of the proposed technique. For the usage of such enhanced similarity measures for social media data within tex-

tual CBR, we first need to populate case bases with data from social media. Since social media data do not readily come in the form of problem-solution pairs, extracting problem-solution pairs from social media data is a topic of interest for research. Though we do not address the problem of building cases from general social media data in this thesis, extracting solutions from social media data such as discussion fora has been experimented with reasonable success lately (e.g., Catherine *et al.* (2013); Gandhe *et al.* (2012); Catherine *et al.* (2012)). This is in addition to efforts such as ExperienceWeb that focus on reusing user experiences from the web in case bases (Smyth *et al.*, 2009; Plaza, 2008).

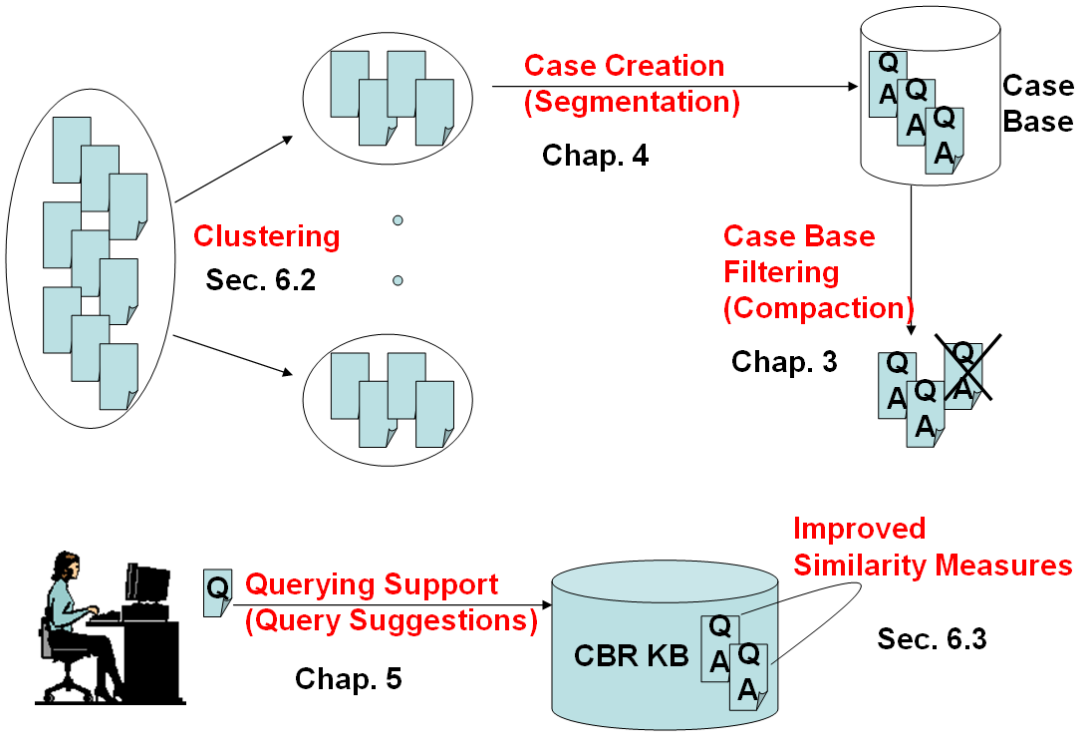

Figure 1.7: Overview

1.4.4 Putting it all together

Having described the various challenges in enhancing Textual CBR systems using general text data, we now illustrate how the various techniques described in this thesis would help towards the goal. An overview of a system that employs our techniques towards exploiting text data in Textual CBR system appears in Figure 1.7. After drilling down to domain specific documents of interest using interpretable clustering, the segmentation technique is applied on problem-solution documents within the domain, to

derive cases. These cases are fed into the case base, and periodically filtered using compaction techniques. The user of the Textual CBR system is then assisted by the query suggestion mechanism, with the similarity between microblog data in the case base assessed using the improved similarity measures that we propose.

1.5 Thesis Overview

In this chapter, we introduced the various challenges in exploiting unstructured text data to enrich Textual CBR systems. We illustrated the enormous growth of data in recent years and briefly summarized the advances in text mining that could be relevant to Textual CBR systems. We then discussed Textual CBR systems, and their difference from other fields such as Information Retrieval and Natural Language Processing. Some of the challenges in exploiting text data to augment Textual CBR systems were then presented under three different themes; *procurement*, *maintenance* and *retrieval*. The technical problems that we address under each of the three themes were outlined briefly.

Chapter 2 describes various concepts from Textual CBR, knowledge management and information retrieval that will be used across the various techniques proposed in the rest of the thesis. Thus, Chapter 2 provides the background material that serves as a context for positioning the techniques presented in this thesis. The techniques that we propose towards improving Textual CBR systems are diverse due to dealing with various stages in the Textual CBR system. We outline how they may be used in a Textual CBR system to enhance the value of such a system; however, they are best evaluated against the state-of-the-art from their respective areas which include data mining, text segmentation, information retreival and case base maintenance. To keep the narrative linear and simple, we evaluate each of the proposed techniques against the state-of-the-art from the appropriate field, in the respective chapters itself. The experimental setups and datasets that are shared across the various evaluations are introduced in Chapter 2.

Following the background section, we first address problems that deal with data in the form of textual problems and solutions, and describe various techniques built towards addressing such problems. Chapter 3 describes the work related to compaction of textual case bases. In Chapter 4, we discuss the technique for segmentation of two-part text documents into cases.This includes techniques for quantifying usability of a

problem to a solution, and using such usability estimates to derive compacted case bases. Chapter 5 describes the improved query suggestion technique for textual case bases.

Chapter 6 covers problems that deal with text not necessarily in the form of textual problems and solutions. This includes the pre-filtering problem of using clustering to drill down to a domain-specific text dataset, and that of improved similarity metrics for the noisy text data that are common in sources such as community-driven question answering datasets. We conclude the thesis with a list of major contributions in Chapter 7. We also list various avenues of possible future work towards furthering our goal of improving Textual CBR systems therein.

CHAPTER 2

BACKGROUND

2.1 Introduction

As alluded to in the previous section, the advent of the Internet Age has resulted in creation of a vast amount of unstructured data. In this work, we are interested in leveraging reusable information from unstructured data generated on the web and the enterprise within the Case-based Reasoning framework. Organizations involved in delivering services often produce numerous textual reports; examples include medical diagnosis reports and problem ticket descriptions with associated root-cause-analysis reports. The piling bunch of historical information such as these have opened up numerous possibilities of knowledge reuse in such sectors. For example, a customer service agent who is posed with a new customer complaint would be able to make good use of solutions to similar complaints that have been encountered in the past. The work presented in this thesis deals mostly with issues on enabling the CBR framework for exploiting historical unstructured text data to solve new problems.

This section briefly introduces problem solving using Case-based Reasoning with a brief discussion about CBR applied to textual data. We then discuss certain aspects of Textual CBR that are pertinent to the work presented in this thesis, positioned with respect to earlier work.

2.2 Case-based Reasoning

Case-based Reasoning (CBR) is a framework for problem solving that focuses on reusing solutions of similar past problems in solving a new problem (Kolodner, 1992). The main underlying premise in Case-based Reasoning is that similar problems have similar solutions. As discussed in Chapter 1, the CBR process has four main phases: *retrieval*, *reuse*, *revise* and *retain*. An illustration of the main processes in CBR appears in Figure 2.1. The pairs $[P_i, S_i]$ represent problem-solution pairs that are held in the case base.

When a new problem P is encountered, top-k similar problems, denoted as P_1 through P_k, are *retrieved*. Their solutions S_1 through S_k are *reused* to form a new solution S that is purported to be usable for P. However, necessary modifications may be applied to *revise* S to form a more appropriate solution, S^* by the user. The $[P, S^*]$ pair is then *retained* among the other pairs in the case base if it is expected to help the case base in problem solving.

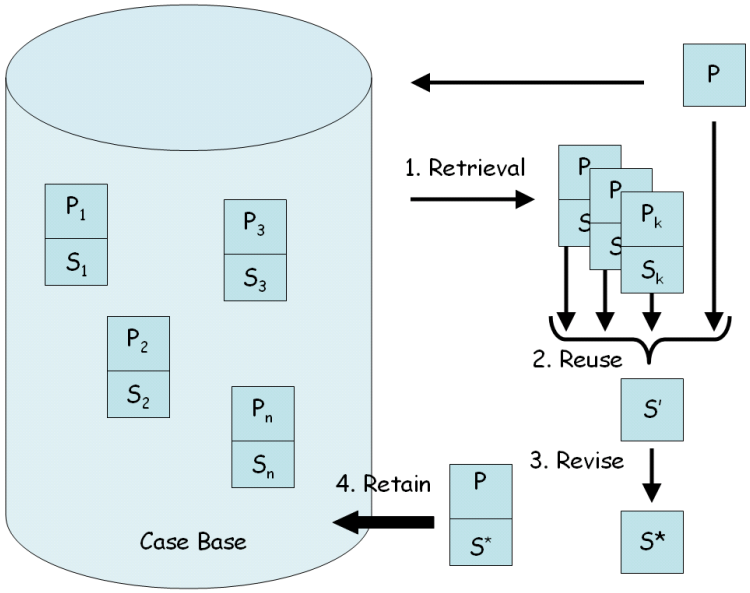

Figure 2.1: The Case-based Reasoning Flow in Schematic

Example: Consider a case-based system that maintains a case base consisting of problems represented as $\{source, destination\}$ pairs and solutions that describe the precise route to be followed. When posed with a travel request specified as 2-tuple in $\{source, destination\}$ format, appropriate travel plans would be output as recommendations. For a new problem that queries for a route plan from A to B, retrieved cases may have route plans from A to B' (where the latter is proximal to B) and those from B' to B. Such route plans may be collated to re-use for the new problem. Upon application of the formulated solution, small improvements such as avoiding an unnecessary detour may be found, and those could be used to revise the route plan, followed by which the revised plan may be submitted along with the request to the case base, for usage to solve requests in the future. ■

2.2.1 Textual Case-based Reasoning

Textual CBR (Lenz *et al.*, 1998) is the application of CBR to scenarios where the problems and/or solutions in the case base are expressed in text. Among the four phases in Textual CBR systems, *reuse* and *revise* phases may be understood to be the more difficult ones since modifying solutions often involves creation of new natural language content; the associated task of natural language generation has been recognized to be hard, even for domain-specific applications (Harris, 2008). Recently, there have been some advances in text reuse (Adeyanju *et al.*, 2010, 2009; Lamontagne and Lapalme, 2004). On the other hand, *retrieval* has been heavily addressed in Textual CBR literature. Though the integration of domain knowledge and any available structured data were touted to be the main differentiators that distinguish Textual CBR from Information Retrieval (Lenz, 1998), many specialized text manipulation techniques from the Information Retrieval community have looked at exploiting the problem-solution dichotomy in textual cases. For example, the lexical chasm (i.e., disconnect) between the problem and solution parts has provided interesting new opportunities for retrieval (Berger *et al.*, 2000; Xue *et al.*, 2008).

Table 2.1 shows a couple of sample textual problem-solution pairs. The first one is from a report from an airline company where the problem part describes the symptoms of the problem as observed by the airline staff. The solution part narrates the steps undertaken to solve the problem. A similar two-part structure is observed in medical reports, like in Example 2, where the "problem" part describes the symptoms, and the solution part describes the *diagnosis* which could be one or more possible diseases that gave rise to the symptoms. Though calling the list of diseases as the *solution* part may sound odd, the diagnosis indeed corresponds to the solution that is sought when experiencing a disease and consulting a physician.

2.2.2 Question Answering and Textual CBR

Question Answering (Kwok *et al.*, 2001) is a discipline in the intersection of Information Retrieval and Natural Language processing, which at a high level, is remarkably similar to Textual CBR. Superficially, both accept questions from the user and produce answers in response. However, the methods adopted for problem solving among these

Example 1	**Problem:**	A Nexen employee checked his rebreather unit and found it to be past its "Service Due Date". He alerted the other passengers and they found the same situation with their re-breather units.
	Solution:	All the units were promptly changed by the heliport staff and the flight proceeded as normal.
Example 2	**Problem:**	School reports continuing difficulties with repetitive questioning. Obsession with cleanness on a daily basis. Inability to relate this well in the classroom.
	Solution:	Asperger Disorder. Obsessive Compulsive Disorder.

Table 2.1: Sample Textual Cases

disciplines differ much, making these very different fields of scientific exploration. We now describe the major differences.

The differences start with the nature of the knowledge used by the different techniques. Textual CBR requires a set of problem-solution pairs whereas any form of structured or unstructured knowledge bases are usable in question answering; the IBM Watson Question Answering system makes plentiful use of Wikipedia knowledge (Ferrucci *et al.*, 2010). Secondly, instead of searching for similar problems (in order to solve a new problem), Question Answering systems use NLP techniques to analyze the kind of question, and perform semantic analysis as to what information the question is seeking. Question Answering systems often are built to handle to-the-point questions that solicit short phrases as answers, such as *"On what day did Christmas fall last year?"*; on the other hand, Textual CBR works with questions that require detailed step-by-step responses such as *"How do I install a wi-fi router with a linux machine?"*. The central hypothesis of Textual CBR, i.e., *similar problems have similar solutions*, doesn't necessarily hold for Question Answering since they deal with short questions and even shorter answers; there is often not enough redundancy to meaningfully use similarity measures to even test or use the CBR assumption. Most of the effort spent in Question Answering systems is in *extracting* the correct answer from the knowledge base, whereas the complexity of Textual CBR systems are in the retrieval phase of finding similar questions and presenting their answers, with the *reuse* phase (that may be seen as analogous to the *extraction* phase in question answering) being at least partially left to the user due to the difficulties involved in composing text from multiple solutions.

Thus, the fields of Textual CBR and question answering have very few tasks in similar, and techniques for each have evolved very differently.

2.3 Procuring Case Bases for Textual CBR

Case acquisition or procurement has been very well-studied for CBR systems that deal with structured data where use guidance has been found to be useful (Shokouhi *et al.*, 2010; Jantke and Dotsch, 1997). Case acquisition for CBR systems that work with textual data has also been looked at (Badra *et al.*, 2009; Proctor *et al.*, 2005) though not as extensively. In order to make use of the vast amounts of data in the form of unstructured text documents in Case-based Reasoning systems, one needs to drill-down to domain specific collections that are relevant to the domain of the Textual CBR system in question. Thereon, there are two intuitive problems that need to be addressed to extract cases from unstructured text data.

- **Filtering Documents:** The large majority of unstructured text documents that are available on the web may be unsuitable for Textual CBR since they do not have a problem-solution structure. These include event descriptions such as newswire reports (e.g., Reuters data[1]), corporate emails (e.g., Enron dataset[2]), newsgroup postings (e.g., 20-NewsGroups data[3]) and so on. Thus, filtering large document datasets to arrive at the subset of documents that are *likely* to be suitable for Textual CBR is a necessary pre-processing step towards making the knowledge in text documents available for Textual CBR systems.

- **Identifying Problem-Solution Parts:** Having identified text documents that are likely to have the problem-solution structure that can be harnessed by textual Case-based Reasoning systems, we still would have to identify the problem and solution components from those documents to build cases.

In this section, we will provide an overview of techniques in literature that could be seen as candidates to tackle the problems outlined above. Though the discussion may

[1]http://www.daviddlewis.com/resources/testcollections/reuters21578/ accessed February 5^{th}, 2014
[2]http://www.cs.cmu.edu/~enron/ accessed February 5^{th}, 2014
[3]http://qwone.com/~jason/20Newsgroups/ accessed February 5^{th}, 2014

not cover all techniques that could possibly be employed for the above, we do attempt to cover the more popular ones in literature that could be used off-the-shelf or can be adapted easily for the same.

2.3.1 Filtering Documents

Our task of identifying a subset of documents that include textual problem and solution descriptions (apart from containing other kinds of textual narratives) could be cast as either a classification problem or clustering problem depending on the availability of labeled data. We will briefly discuss our problem of filtering out case-like documents from both these perspectives below.

Classification for Filtering: In the presence of a set of documents which are labeled as *desirable* for Case-based Reasoning, and another set that is not, a classification technique would build a mathematical model that captures the differences in nature between the *desirable* documents and others. The two categories are referred to as classes, in classification literature, and our setting is similar to that of *spam filtering* where emails are to be classified as either spam or not (Padmanabhan *et al.*, 2006). The learnt classification model can then be applied on new documents to ascertain whether they belong to the *desirable* class or not. Towards applying classification, a document representation is first chosen; words contained within the document are often used as features (or attributes) for the document representation. The normalized frequency of each word in the document is used as the value of the attribute, for that document. Various classification models may be learnt over such document representations; options include decision trees (Quinlan, 1986) that create a hierarchical division of the data space using conditions on attribute values, Support Vector Machines (Joachims, 1998) that learn a decision boundary between documents belonging to different classes, or generative models such as Naive Bayes (Ng and Jordan, 2001) that learn models on how words are picked for documents in each of the classes. The success of text classification has mostly been in domains where the task is to separate documents into domain-specific classes (Chakrabarti *et al.*, 2003) such as topics in the Reuters newswire articles. In our problem, since the classes differ in whether they contain problem-solution parts, the classification presumably would have more to do with *writing style* and presence of some key words like question-signifying words (e.g., *what*, *how* etc.); this makes the

classification problem harder.

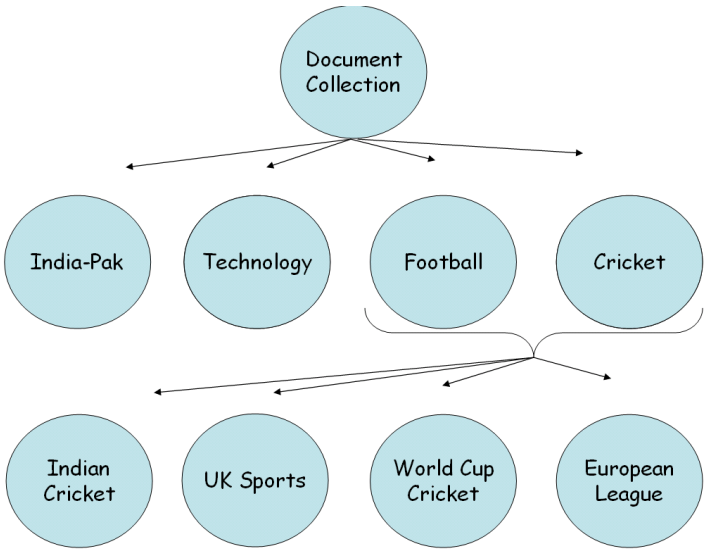

Figure 2.2: Scatter Gather Example

Clustering for Filtering: Scatter Gather (Cutting *et al.*, 1992) was among the first approaches for using clustering as a tool to drill-down to document subsets. This technique produces clusters with a (collection of) word(s) typifying each cluster, wherein the user can select one or more clusters and expand them into a larger set of clusters. An example usage of Scatter Gather is shown in Figure 2.2 where each of the large ovals represent a cluster of documents. At each step, one or more of the clusters are chosen to *explode* into the next set of clusters. Techniques such as K-Means and Hierarchical Agglomerative Clustering (Jain *et al.*, 1999) or more recent clustering techniques could be used as tools to apply at each such interactive clustering step. In our problem of identifying subsets of documents that are desirable for CBR, it would be intuitive to make use of cluster descriptions such as sets of words that typify a cluster; the presence of words representative of problems and solutions in those descriptions could provide strong hints on the presence of desirable documents in the respective cluster. However, problem words and solution words are often very domain-specific. For example, *disorder* could be a common solution word in a medical diagnosis system whereas its usage in an airline troubleshooting system could be in a generic sense and not specialized to the problem or solution part. In the interest of developing cross-domain solutions, we do not explore automatic methods for the filtering problem in this work.

2.3.2 Identifying Problem-Solution Parts

The set of documents deemed to contain problem and solution text, as identified using the filtering steps above, would now have to be processed so that the problem and solution segments can be extracted from them. Each of these documents could potentially have more information than just the problem and solution text excerpts; thus, all information other than the problem and solution text may be discarded to create one or multiple cases from each of the documents.

Text Segmentation deals with the problem of breaking up text documents into coherent segments. We expect to deal with domain specific problems (solutions) that would consist of *coherent* sentences where text segmentation could be used as a first step towards addressing the problem of case creation from documents; segments could later be labeled as being either *problem* or *solution* or *other*. Early text segmentation algorithms such as TextTiling (Hearst, 1994) segment text documents into lexically coherent segments using lexical frequency and word distribution information. Later techniques for text segmentation focused on using word repetitions (Reynar, 1994), semantic networks (Kozima, 1993) and deep semantic features such as co-reference that encompasses a candidate segment boundary (Passonneau and Litman, 1997). A very recent text segmentation technique uses affinity propagation, a clustering-like technique, to learn segment boundaries (Kazantseva and Szpakowicz, 2011). It may be noted that the text segmentation algorithms are oblivious to the label to be assigned to each text segment (such as *problem*, *solution*, or *other*), and simply deal with the problem of splitting text into coherent pieces.

In our setting, we are likely to have many candidate documents that belong to the same domain (e.g., a collection of reports from an airline domain, or a collection of diagnosis reports from the medical domain); thus, usage of techniques that can leverage collection-level statistics to segment each document better could be appropriate. Kummamuru *et al.* (2008) presents a technique that uses a collection of call transcripts to achieve higher segmentation accuracy than by dealing with each document in isolation. Though we have not come across unsupervised segmentation algorithms that assign a *label* (e.g., *problem* or *solution*), segmentation of text with the aid of extrinsic structured data banks (e.g., EROCS (Chakaravarthy *et al.*, 2006)) yield an intuitive label for each segment, the entity to which the segment best relates.

Having segmented the text into multiple coherent segments using techniques such as those summarized above, we still need to identify the problem and solution segments. If the label of each segment is not automatically derived from the segmentation, we would have to resort to techniques that use supervised data to learn multi-class classification models that can label each segment into either of *problem*, *solution* or *other* buckets; upon such labeling, problem-solution pairs may be easily constructed. We will address a specialized version of the problem-solution identification problem, that of segmenting text documents that are known to contain just two segments (problem and solution), in a later section.

2.4 Maintenance of Textual Case Bases

While case bases maybe enriched with new cases as part of the *retention* phase whereby newly posed problems are retained along with their revised solutions, allowing case bases to grow in uncontrolled fashion has not been seen as entirely desirable. The *utility* problem in case-based systems (Smyth and Cunningham, 1996) suggests that the efficiency of a case based reasoning system would degrade when the size of the case-base increases. This realization sparked much research on maintenance of case bases where cases are periodicially filtered out; this process is referred to as *case base maintenance* or *case base mining* in CBR parlance. Proposals to tackle the utility problem differ on the methods adopted to limit the number of cases in a case base by discarding some cases; the differences could be in the criteria used for discarding and the algorithmic details (e.g., decremental, incremental, greedy etc.) of the compaction process.

Early research on limiting the training set focused on using classification accuracy as the criterion to select cases to remove (in a decremental approach (Kibler and Aha, 1987)) or add (in an incremental approach (Aha and Kibler, 1991)). With solution parts of cases getting increasingly sophisticated and diverse, *coverage* and *reachability* (Smyth and Keane, 1995) became more accepted measures of selecting cases to preserve. Coverage of a case refers to the set of cases that (the solution part of) that case can solve; reachability of a case is the set of cases that could solve the (problem part of the) case. Formally, for a case base $C = \{c_1, c_2, \ldots, c_n\}$,

$$Coverage(c) = \{c' | c' \in C \wedge Adaptable(c, c')\}$$

$$Reachability(c) = \{c' | c' \in C \wedge Adaptable(c', c)\}$$

where $Adaptable(x, y)$ denotes whether the solution part of x can be used to solve the problem part of the case y.

Further, a later work (Smyth and McKenna, 1998) emphasizes that such competence models should be retrieval-conscious; it is not just enough to have a case (in the case base) that could be used to solve a new problem, but that such a solution case should be among the ones retrieved by the retrieval engine for the new problem. For every compacted case base, the competence is evaluated as the number of cases from the test set that cases within the compacted version can solve. This process may be seen as analogous to the train-test model in machine learning methods where the compacted case base may be seen to be similar to the trained model in supervised machine learning. In cases where a test set is not available, competence can be measured against one or multiple held out set of cases such as is commonly used in cross-validation (Kohavi, 1995). Generalizing the concepts in Smyth and Keane (1995), competence may be formally outlined as:

$$Competence(C, T) = \bigcup_{c \in C} \{t | t \in T \wedge Adaptable(c, t)\}$$

Formally, the competence of the case base C with respect to the test set, T, of cases, is thus computed as the set of cases in T that can be solved by one or more cases in C. Case base mining is also used to refer to the task (Pan *et al.*, 2007) of trading off between the often contradictory goals of removing more cases and retaining competence.

As previously illustrated, Case-based Reasoning has traditionally focused on structured cases like a *[source, destination]* specification for a travel planning system (Zhu and Yang, 1999) where the solution is a plan for traveling from the source to the destination. Whether a solution *solves* a particular problem may even be analyzed automatically in such cases; the quality of the solution could be assessed using metrics such as total cost of the route. Unlike this, in Textual CBR, there are no easy ways of automatically determining whether a solution solves a particular problem. However, the

advancements in statistical processing of text from fields such as text mining have led to various standard methods of computing similarity between text segments. Among the more popular similarity estimation methods for text data is the cosine similarity metric computed over tf-idf (Wu *et al.*, 2008) vectors which are essentially weighted word-frequency vector representations of text segments. This opens up possibilities of leveraging such text similarity metrics along with extrinsic quality estimates (such as the number of votes accrued towards a solution) to estimate *usability* of textual solutions to problems in textual case bases. Such usability estimates may be used for improving case acquisition and/or maintenance techniques for Textual CBR. We will elaborate on text similarity metrics and its usage in CBR *retrieval*, in the next section.

2.5 Retrieval for Textual CBR

Retrieval is the first and often the most important phase in Case-based Reasoning since it involves the selection of the subset of cases that are used in later processes, towards solving the problem posed by the user. We briefly summarize the retrieval process for classical CBR where the cases are structured and go on to describe various measures of similarity for textual data, from fields such as data mining and information retrieval.

2.5.1 Classical CBR Retrieval

Consider a CBR system for the automobile sector; the problem part could contain a brief textual description of the problem followed by values for various attributes for the automobile in question, whereas the solution part may consist of a diagnosis of the problem, and suggestions for repair. When a new problem is posed against such a CBR system, the similarity of the problem to (problem parts of) cases in the case base may be estimated as an aggregate of the similarity of the values in the various attributes. As illustrated in Figure 2.3, the similarities between the values of the individual attributes are first computed and then aggregated (e.g., using aggregation functions such as *sum*, *min*, *max* etc.) to form a single value to denote the similarity between the new problem and the chosen case. The similarity measure used for different attributes could be different; numeric similarity measures may be used for numeric attributes, and text similarity measures could be employed for textual attributes.

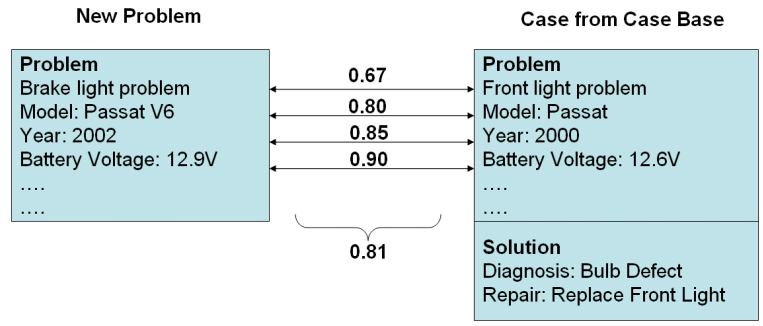

Figure 2.3: Example Similarity Computation for CBR

Once such a similarity score is estimated for the (problem parts of) cases in the case base against the new problem, the top-k cases that have the highest similarity values are retrieved by the engine in most scenarios. Indexing techniques are often employed to avoid computing similarities against all cases in the case-base so that retrieval is fast enough (Stéphane *et al.*, 2010).

2.5.2 Representing Text Data

Text data comprise a sequence of words, interspersed with whitespaces, sentence delimiters and other punctuation. However, the most popular and effective text representations disregard the order or words appearing, and view the text document as a bag of words. Thus, the two text segments *"the problem is with the front light"* and *"the light with problem is the front"* are indistinguishable in this representation. For each text segment and for every word that appears within it, information retrieval methods assign a non-negative weight; this leads to representing a text document as a vector of weights. The simplest representation scheme uses boolean representations whereby the wieght is unity if the word is present and zero if absent. For notational convenience, only the non-zero weighted terms are represented, thus, leading to the following representation for the text segment *the problem is with the front light*:

$$\{the : 1, problem : 1, is : 1, with : 1, front : 1, light : 1\}$$

The term frequency vector model assigns a value for each attribute that corresponds to the frequency of the word in the document. This leads to a slightly different representation as below:

$$\{the : 2, problem : 1, is : 1, with : 1, front : 1, light : 1\}$$

However, it may be intuitive that all words do not convey the same amount of information. For example, words like *the* are far less informative than, say, *light*, for the above example. The popular method to account for such considerations is to weigh the terms (i.e., words) by a word-specific weight knows as *inverse document frequency* (Robertson, 2004). However, before applying *idf*, it is often the convention to normalize the term frequency vector so that the values add up to 1.0. The normalized term-frequency vector would be:

$$\{the : 0.29, problem : 0.14, is : 0.14, with : 0.14, front : 0.14, light : 0.14\}$$

The *idf* weight for a term is a collection-level statistic that is estimated using a collection of documents. Let the collection of documents be $\mathcal{D}$, the *idf* of a term w is then:

$$idf(w) = log\ \frac{|\mathcal{D}|}{|\{d|d \in \mathcal{D} \wedge w \in d\}|}$$

Now, the *tf-idf* vector for a document is constructed by assigning a value for each word that is the product of the normalized term frequency weight, and the *idf* corresponding to it. Formally,

$$tf\text{-}idf(w, d) = \frac{f(w, d)}{\sum_{w' \in d} f(w', d)} \times idf(w)$$

where $f(w, d)$ denotes the frequency of the word w in the document d. For example, if the word *the* occurs in 50% of the documents, the *idf* weight would be $log(2) = 0.3$; this makes the *tf-idf* weight for *the* in our example text fragment example to be $0.29 \times 0.3 = 0.087$. Most common document representations in information retrieval and data mining follow the *tf-idf* or similar models for representing text documents.

2.5.3 Computing Text Similarity

With documents represented as *tf-idf* vectors, the similarity between a pair of documents would then be computed as the similarity between their *tf-idf* vector representations. While information retrieval concerns itself mostly with comparing documents against short query phrases, text clustering is a field in data mining that deals with comparing whole documents. The most common similarity measure between pairwise *tf-idf* vectors used in clustering is the Cosine Similarity measure (Huang, 2008).

$$CosineSim(d_1, d_2) = \frac{\sum_{w \in d_1 \vee w \in d_2} tf\text{-}idf(w, d_1) \times tf\text{-}idf(w, d_2)}{\sqrt{\sum_{w \in d_1} tf\text{-}idf(w, d_1)^2} \times \sqrt{\sum_{w \in d_2} tf\text{-}idf(w, d_2)^2}}$$

The cosine similarity as illustrated above computed the angle between the normalized unit vectors corresponding to each of the documents. Other similarity measures such as Jaccard Similarity[4] and KL-divergence[5] could be used instead.

2.5.4 Retrieval Models in IR

The fundamental task in information retrieval is that of ranking documents in a collection with respect to a query that is posed by the user. The user query often contains very few words; for IR systems that work on web data, the query length has been found to be 2.4 words on the average (Spink *et al.*, 2001). Though classical CBR deals with fully specified problems and thus leads to scenarios analogous to document comparison, Textual CBR may encounter the case that the new problem is specified very minimally using very few words, leading to IR-type scenarios.

Okapi BM25 (Manning *et al.*, 2008) is an early and still popular technique for ranking documents with respect to a query. For a query Q containing words $\{q_1, q_2, \ldots, q_k\}$, the score of a document d according to a BM25 instantiation, may be computed as:

$$Score(Q, d) = \sum_{q \in Q} idf(q) \times \frac{f(q, d) \times (k_1 + 1)}{f(q, d) + k_1 \times (1 - b + b \times \frac{|d|}{avgdl})}$$

[4] http://en.wikipedia.org/wiki/Jaccard_index accessed February 5^{th}, 2014

[5] http://en.wikipedia.org/wiki/Kullback-Leibler_divergence accessed February 5^{th}, 2014

where $|d|$ and $avgdl$ denote the number of words in d and the average number of words across documents respectively, and k_1 and b are parameters to the ranking function. The usage of document length in the denominator prioritizes documents that have a larger fraction of words overlapping with the query.

2.5.5 Retrieval Models for Problem-Solution Repositories

Information Retrieval over corpora of textual documents that encompass a problem and solution part has attracted attention from the IR community off late. *Lexical chasm* (Jeon *et al.*, 2005), the difference in the lexical behavior between problem and solution parts, is of concern while processing problem-solution collections. The query phrase may be intuitively expected to be lexically similar to the behavior of problems in the corpus whereas the user expects retrieved results to contain relevant solutions as well; this poses interesting challenges for retrieval. Statistical machine translation models such as IBM Model 1 (Brown *et al.*, 1990) can be used to learn the lexical correlations between the problem and solution parts by treating the collection as a parallel corpus of problems and their solutions. Such an approach was proposed by Xue et. al.(Xue *et al.*, 2008) where the probability of generating the query from a document is used as a score of relevance of the document to the query. The simplified formulation that captures the crux of the technique (while excluding smoothing aspects) is as follows:

$$P(q|d) = \prod_{w \in q} \left[\alpha P_{ml}(w|p(d)) + \beta \sum_{t \in p(d)} P_T(w|t) P_{ml}(t|p(d)) + \gamma P_{ml}(w|s(d)) \right]$$

where $p(d)$ and $s(d)$ denote the problem and solution parts in document d, and the different weighting parameters are denoted by α, β and γ. $P_{ml}(w|s)$ denotes the maximum likelihood estimate (MLE) (Zhai and Lafferty, 2001) in generating the word w from the text segment denoted as s. The usage of translation models is in the second term in the right-hand-side of the equation, where the probability of generating each query word, w, from d is estimated through words in $p(d)$ using a translation model; for each word t in $p(d)$, the probability of generating w from t as assessed by the translation model (i.e., $P_T(w|t)$) is weighted by the maximum-likelihood estimate of t itself, and

factored into a summation.

Conceptually, the above formulation due to Xue *et al.* (2008) may be seen as expanding the query using correlated words from the translation model, followed by using traditional techniques such as MLE.

2.5.6 Query Suggestions to aid Retrieval

With web search engines becoming the primary means of finding information on the web, interactive real-time support for querying has become ubiquitous. Query suggestions are a popular feature towards providing user with support for querying over large text collections. Figure 2.4 illustrates the operation of Google's query suggestion feature; when the user has entered some text in the query box which could comprise a few complete words followed by an optional incomplete word, the real-time feature is triggered and shows the user various possible completions for the query. An early study on the effectiveness of query suggestions reports 30% uptake (Feuer *et al.*, 2007) of query suggestions.

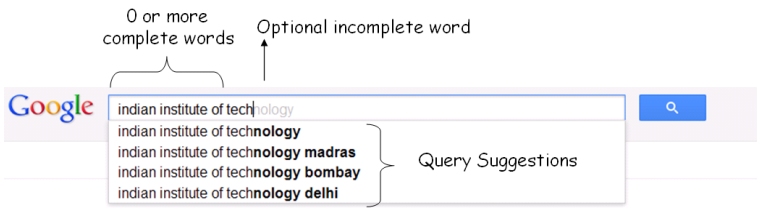

Figure 2.4: Google's Query Suggestion in Operation

Query suggestions deal with the problem of suggesting completions to an incompletely specified information need, and hence is different from the problem of interactive query refinement (Fonseca *et al.*, 2005) that starts with a completely specified query as input and attempts to discover meaningful specializations. Thus, *world c* is a potential candidate input for query suggestion techniques that could attempt to complete the second word to form queries such as *world cup*, whereas query refinement techniques may be employed to refine queries such as *world cup* to possible specializations such as *football world cup*. Since textual queries have not been the focus of traditional CBR systems, literature in query suggestions has mostly originated from the information retrieval and knowledge management communities.

Query suggestion techniques have extensively used query logs and such usage data (Jones

et al., 2006). The intuitive method is to use complete queries from history as suggestions for new incomplete queries. Usage of context information from click-through data and session information to enable context-aware query suggestions (Cao *et al.*, 2008) has also met with good success. Pre-processing historical queries by linking related queries to create a query flow graph (Boldi *et al.*, 2009) and suggesting queries that are reachable by short random walks from the incomplete query has been shown to improve query suggestion effectiveness. Similarity between queries may alternatively be estimated based on landing page correlations (Cucerzan and White, 2007). Usage of learning approaches (Santos *et al.*, 2013) and personalization of query suggestions (Shokouhi, 2013) have been of recent interest in improving query suggestion systems.

However, the above methods that rely on usage information are mostly not applicable to domain-specific search systems due to a small user base, leading to unavailability of plentiful usage data. Query suggestion techniques that work in the absence of query logs (Bast, 2006; Bhatia *et al.*, 2011) have recently been developed to address this problem. It may be noted that Textual CBR systems are also likely to be domain specific and hence phrase harvesting from the underlying document corpora would be more applicable to them than techniques that exploit historical usage data. The first technique for deriving query suggestions from non-historical data was CompleteSearch (Bast, 2006) that focuses on finding completions for queries that would generate more results; here, the preference for a candidate query suggestion is considered to be directly related to the number of documents from the underlying corpora that would be treated as relevant (based on, say, a thresholding on a scoring function such as Okapi BM25, for querying over text corpora) to the candidate query suggestion. The technique presented in (Bhatia *et al.*, 2011) improves upon such frequency based scoring by considering the co-occurrence relationships between terms in documents and using such co-occurence scores, in combination with *idf*, to score the candidate query suggestion.

The problem solution partition inherent in textual case-bases opens up new possibilities for the problem of query suggestions to aid retrieval over such corpora. We address the problem of query suggestions over textual case bases in a later section.

2.6 Incorporating Data from Social Media

The emergence of social media has led to an unforseen explosion of real-time interaction data between social media participants. Thus, unlike classical Textual CBR with a mostly static case base, social media based Textual CBR can probably attempt to solve problems by capitalizing on additional metadata that is usually available in social media such as the time of posting, the user who created the content, the reputation of the user and the location of posting (e.g., on a fan page of a public figure, or on the customers page of a product). Social media, through the lens of Textual CBR, may be viewed as being belonging to two classes; one in which the problem-solution structure is explicitly available, and another in which social media content would have to be mined to create case bases. Examples of the former include community question answering (CQA) websites like Yahoo! Answers[6] and Quora[7], whereas Facebook[8] and Twitter[9] are of the latter kind.

Textual CBR on CQA Data: Community-driven Question Answering systems are manifestations of knowledge markets on the internet. Unlike Question Answering systems referred to in Section 2.2.2, solutions from CQA systems are often descriptive and are hence mostly reusable like those dealt with in Textual CBR systems. The Among the earliest ones that used a monetary reward as an incentive mechanism to encourage solution authors include Experts-Exchange[10], the now defunct Google Answers[11] and Uclue[12]. On the other hand, Yahoo! Answers and Microsoft Live QnA[13] are free knowledge markets, whereas the recent Quora[14] uses some form of virtual currency called Quora credits. While search mechanisms on such websites are rudimentary with most of them allowing the user to query the data using IR-style mechanisms, all of these have a question-answer structure and may be readily used in Textual CBR. The additional metadata available with them such as user reputation of the solution author and the tags used in the solutions could all be used to enhance efficiency in the retrieval and reuse phases of Textual CBR.

[6]http://answers.yahoo.com accessed February 5^{th}, 2014
[7]http://www.quora.com accessed February 5^{th}, 2014
[8]http://www.facebook.com accessed February 5^{th}, 2014
[9]http://www.twitter.com accessed February 5^{th}, 2014
[10]http://www.experts-exchange.com accessed February 5^{th}, 2014
[11]http://en.wikipedia.org/wiki/Google_Answers accessed February 5^{th}, 2014
[12]http://uclue.com accessed February 5^{th}, 2014
[13]http://qna.live.com accessed February 5^{th}, 2014
[14]http://www.quora.com accessed February 5^{th}, 2014

Textual CBR on General Social Media Data: To be able to incorporate data from non-CQA websites that have content that are not categorized as problems and solutions, the challenges are similar to that described in Section 2.3. However, of particular interest is whether social media is effective in providing useful information for seekers of information. Plaza (2008) argues for exploiting experiential knowledge from the web in CBR systems. There have been various efforts to adapt CBR techniques to handle user-generated content on the web from blogs, wikis (e.g., (Smyth *et al.*, 2009)) and product reviews (Bridge and Healy, 2012). Coming to social media websites, it was found in a recent study that Facebook (Morris *et al.*, 2010) and Twitter (Paul *et al.*, 2011) work pretty well as an information seeking platform; the percentage of questions answered were found to be more than 70% in both. It was also found that the success rate in eliciting responses is correlated with various factors such as conciseness of the question and the inclusion of question marks (Teevan *et al.*, 2011). An interesting characteristic of social media has been that *weak connections* are seen to provide more valuable information to information seekers (Gray *et al.*, 2013); this is so since strong connections typically share the same information sources with the information seeker and hence are likely to have the same perspective on the issue. However, opinion has been divided on this issue with a recent work disputing such correlations between tie strength and information (Panovich *et al.*, 2012). Despite such work on information seeking behavior in social media, interest in social media from a Textual CBR perspective has been scanty. Some of the open issues with respect to applicability of Textual CBR on social network data could be listed as follows:

- *Reusability of Solutions:* The nature of information provided to information seekers in social media is not well known; it could be somewhere in between to-the-point solutions (like in the case of Question Answering - Ref. Section 2.2.2) or reusable solutions as is mostly in the case of CQA data. The former extreme is unfavorable to Textual CBR.

- *Textual Similarity on Short Texts:* Much of social media content is in the form of short text segments. In some cases like Twitter, an upper limit on the length is enforced. It is not clear whether textual similarity measures on traditional text data (like the cosine similarity metric on tf-idf vectors) would work well on such

data. This is of concern since *retrieval*, among the major phases of Textual CBR, relies heavily on the similarity measure.

- *Temporal Relevance:* Most social media are used to share real-time information, and thus the reusability of the solutions may have a temporal dimension as well. Factoring in such temporal dimensions into Textual CBR could help enable better knowledge reuse.

Based on previous literature, it may be inferred that incorporating data from social media into Textual CBR is a relatively unexplored area and there are many challenges to be addressed before social media data may be exploited effectively within the Textual CBR framework.

2.7 Datasets

In this section, we discuss various datasets that we use for empirically evaluating the techniques presented in this thesis. The datasets come from various sources such as general text document datasets, problem-solution datasets, and social media datasets.

2.7.1 Datasets with a Problem-Solution Separation

We now describe the various datasets that we use which comprise documents with a known problem solution separation. CQA datasets that fall under this category have multiple solutions per problem, since multiple users may author solutions for the same problem. The dataset statistics are summarized in Table 2.2. The datasets under the type FAQ were extracted from frequently asked questions pages in various websites, and have been categorized under the domain to which the website belongs. The FAQ datasets were initially collected as part of an Information Retrieval evaluation task and are publicly available[15]. The CQA datasets, *lincoln* and *delhi* and were collected by crawling Yahoo! Answers[16] and the name of the category has been used as the name of the dataset. *lincoln* contains data relating to lincoln cars, whereas *delhi* is the category

[15]http://www.isical.ac.in/~fire/faq-retrieval/2013/faq-retrieval.html accessed February 5^{th}, 2014
[16]http://answers.yahoo.com accessed February 5^{th}, 2014

Type	Name	#Docs	#Solutions	#Solutions per Problem
FAQ	visa	511	511	1
	career	938	938	1
	insurance	990	990	1
	health	433	433	1
	sports	357	357	1
	agri	229	229	1
	loan	514	514	1
	tourism	134	134	1
	railways	268	268	1
	telecom	103	103	1
	web	109	109	1
CQA	lincoln	1403	6591	4.70
	delhi	1534	7940	5.18
Misc	encarta	1126	9839	8.74

Table 2.2: QA Datasets

people use to seek information pertaining to the *new delhi*, the Indian capital city. These have upto 5 answers per question. The other dataset, *encarta*, is a Question Asnwering dataset released by Microsoft Research[17] that, unlike other Question-Answering datasets, has long solutions that comprise multiple sentences. The solutions in this dataset are extracts from the Encarta 98 encyclopedia.

2.7.2 Other Datasets

Since the retrieval and case-base procurement tasks are not very dependent on the availability of a problem solution separation, we additionally use general text datasets for evaluating such techniques. Towards this we use various clustering datasets and social media datasets as summarized in Table 2.3. The text clustering datasets are those used in a previous work on hierarchical document clustering (Zhao and Karypis, 2002). The social media dataset was obtained from Fundacion Barcelona Media[18] and is publicly available; it contains close to one million tweets, and we used user-study based relevance judgements while using this dataset for a retrieval evaluation.

[17]http://research.microsoft.com/en-us/downloads/88c0021c-328a-4148-a158-a42d7331c6cf/default.aspx accessed February 5^{th}, 2014

[18]http://caw2.barcelonamedia.org/node/7 accessed July 6^{th}, 2012

Type	Name	#Docs
Clustering	sports	8580
	k1b	2340
	ohscal3	2864
	r4	1013
	c3	3893
	cranmed	2431
Social Media	twitter	977252

Table 2.3: Non-QA Datasets

2.8 Chapter Summary

In this chapter, we introduced the main concepts behind Case-based Reasoning and its application to textual data. We then described various problems related to Textual CBR that we will deal with, such as *procurement of case bases*, *maintenance* and *retrieval*. We discussed techniques from literature that may be used or adapted for handling such tasks with emphasis on their applicability as well as strength and weaknesses of each technique. The emergence of social media as a way of collaborative knowledge creation has led to various studies on their usability as an information seeking platform. However, the interest in social media from a Textual CBR angle has been minimal in literature; we outlined various challenges in exploiting social media data in Textual CBR systems. Lastly, we concluded with a description of the various datasets that are suitable for Textual CBR research, with special emphasis on those that we will use to empirically evaluate the techniques developed in this thesis.

CHAPTER 3

COMPACTION OF TEXTUAL PROBLEM-SOLUTION DATASETS

3.1 Introduction

Textual problem-solution datasets are increasingly available from Community-driven Question Answering (CQA) portals such as Yahoo! Answers[1] and Microsoft Live QnA[2]. Unlike traditional datasets for Case-based Reasoning, CQA datasets often have multiple solutions per question. The typical operation of Yahoo! Answers, which is among the more popular CQA websites, involves the following steps:

- **Question Posing:** A registered user poses a question on the Yahoo! Answers website; she has the option to conceal her user name.
- **Activity on the Question:** Usually, questions are open and open for answering for *4* days. The user who posted the question, however, can close it after a minimum of one hour, or can extend the activity time to upto *8* days.
- **Points Accrual:** Yahoo! employs a rewards system to reward users who have been active, using a virtual currency called *points*. Points that are accrued by a user is indicative of the quality and quantity of activity in the Yahoo! Answers system. At least *5* points are required to make a user eligible to pose a question.
- **Choosing the Best Solution:** Solutions may be voted on by the members of the community, and once the question is closed for solutions, the community or the question asker would select a *best solution*. The *best solution* author is credited with *10* points; this serves as an incentive to contribute high quality solutions.

[1] http://answers.yahoo.com accessed February 5^{th}, 2014
[2] http://qna.live.com accessed February 5^{th}, 2014

- **Retention:** Most commonly, all solutions posted in response to the question are retained in the question page. The *best solution* is promoted to appear just below the question, and thus gets better visibility.

Quora[3], another CQA system, also follows a similar process as above, and is armed by its own virtual currency called Quora Credits. Currently, the most common way to search the Yahoo! Answers archives is to do a simple Information Retrieval-style search on the website, or using browser add-ons[4]. When a problem is posed on such a system, pages containing similar problems and solutions are then shown to the user who then sifts through the pages to identify and assimilate content of interest. However, since all problems and solutions are available to the IR system, low-quality content is often returned in response to queries[5]. Yet, CQA websites still remain a valuable source of *opinions* and *experiences* whereas *Wikipedia* is mostly for facts.

We conjecture that identifying and retaining the following kinds of information would help in utilizing the abundance of knowledge in QA repositories such as those from CQA websites, in systematic knowledge reuse frameworks such as Case-based Reasoning:

1. **Popular Knowledge:** In the presence of voting systems (like in Yahoo! Answers and Quora), users get to express preferences for certain solutions by voting. Solutions that gather higher number of votes, i.e., the popular solutions, may be intuitively considered more valuable to retain since more users like those. The actual number of votes may be weighted by reputation scores (Resnick *et al.*, 2000) of voters to provide some amount of immunity to spamming.

2. **Generic Knowledge:** Solutions that are *usable* across more problems would be preferred for retention. This criterion tends to prefer generic solutions, and is analogous to overfitting avoidance in supervised learning systems.

3. **Quality:** Solutions that are better phrased, easier to understand, and are more structured (e.g., includes a series of steps rather than a cluttered text description)

[3]http://www.quora.com accessed February 5^{th}, 2014

[4]https://addons.mozilla.org/en-us/firefox/addon/yahoo-answers-search/ accessed February 5^{th}, 2014

[5]http://downloadsquad.switched.com/2006/08/07/yahoo-answers-is-a-disaster/ accessed February 5^{th}, 2014

may be preferred for retention. However, the sophisticated nature of such quality analyses renders it hard to automate and hence quality analysis is best done manually.

Problems and solutions are often referred to as *questions* and *answers* respectively in CQA parlance; so, we will use these terms interchangeably. In this work, we focus on compacting problem-solution datasets by filtering out knowledge based on the first two criteria above. It may seem that larger datasets of historical problem-solution pairs would always lead to a better knowledge reuse system as long as they include highly popular and reusable solutions and the retrieval algorithm is smart enough to ignore the rest at query time. However, the *utility* problem in case-based reasoning refutes such an argument, as pointed out in Section 2.4. Studies say that the effectiveness of the knowledge base often degrades when the size of the knowledge base increases. This led to a series of works on case-base maintenance, that have been summarized in Section 2.4. Though our problem of compaction of CQA data for case base acquisition/maintenance is related to case base maintenance in that both relate to filtering out information, our work differs largely from previous CBR maintenance methods in two significant ways:

- **Multiple Solutions:** As illustrated above, CQA datasets are characterized by the availability of multiple solutions posed for the same problem. Though classical CBR relies on usability of multiple solutions (from across similar problems) to any given problem, it is not very common to have a case-base where there are multiple solutions that are explicitly attached to a single problem; the classical CBR paradigm deals with case-bases that are represented as 2-tuples represented as $[problem, solution]$ pairs. The availability of multiple solutions brings with it new challenges, addressing which lie at the core of the techniques we propose.

- **Textual Cases:** The estimation of *usability* of a solution to a problem is often very domain specific. For example, in a scenario of travel planning, the proximity of the end points of a candidate solution to the end points specified in the problem determines the usability of the solution; the CBR system specialized in candidate recommendation for jobs should match the specific skill sets specified in the job description to the one in the candidate resume. On the other hand, comparing

textual solutions and problems can largely be done using off-the-shelf text similarity metrics. In our problem of compacting textual CQA datasets, we will make generous use of advancements in text similarity computations from Information Retrieval and Natural Language Processing literature in devising techniques.

Since we start off with a dataset that is already known to contain problems and solutions (just like case bases), we will call our problem as *case base compaction*. Given that our problem scenario is different from the ones in literature, as discussed above, we will also outline various evaluation measures for a case base compaction technique that exploits information such as *popularity* (as a proxy for usability) and *genericity* of solutions.

3.2 Motivation

Towards motivating our problem, we now outline some examples, with increasing complexity, leading up to a version of the problem that we address.

Non-overlapping Coverage Spaces: The outer black boundary in Figure 3.1(a) is meant to represent a space of problems. A1, A2 and A3 are solutions available in the case-base; corresponding ovals represent the subset of problems (called the *coverage space*, as introduced in Section 2.4) that they can *solve*. We call a solution as *usable* for a problem if the latter is within the coverage space of the former. As may be seen from the figure, A2 can solve more problems than A1 and A3 due to its larger coverage space, assuming that the distribution of problems in the problem space is uniform. Consider the problem of choosing two solutions from among $\{A1, A2, A3\}$ for building the case base. In this example, we do not have varying degrees of usability; thus, the only criterion among those in Section 3.1 that could be applied is that of genericity (solving as many problems as possible). This makes the optimization problem similar to the *set cover problem* (Cormen *et al.*, 2001), when each solution is represented by the set of problems it can solve. A simple heuristic could greedily choose the solutions with the most coverage, for inclusion. This would choose A2 and A3, since only two solutions were permitted.

Overlapping Coverage Spaces: When coverage spaces among solutions overlap, the greedy choice strategy does not work well, as we will see for the case in Figure 3.1(b), where the greedy solution would choose A4 and A5. Due to the large overlap between the coverage space of A4 and A5, such a choice does not augur well with our rather intuitive intent of preserving solutions that together solve as many problems as possible. A greedy incremental heuristic (Yang and Zhu, 2001) starts from an empty set of solutions, and keeps adding solutions that are useful for the maximum number of problems outside the set of problems already covered by the set of solutions selected so far. If the number of solutions to be chosen is limited to two as in the earlier case, the incremental approach for the scenario in Figure 3.1(b) would choose A4 and A6. This is so since the incremental coverage criterion avoids counting problems that could be solved by both A4 and A5 twice. This could also be achieved by differentially weighting problems; problems in the intersection of A4 and A5 could be weighed half as much as problems that are in the coverage set of only a single solutions. Such weighting coupled with the simple heuristic has also been found to be useful (Smyth and McKenna, 1999).

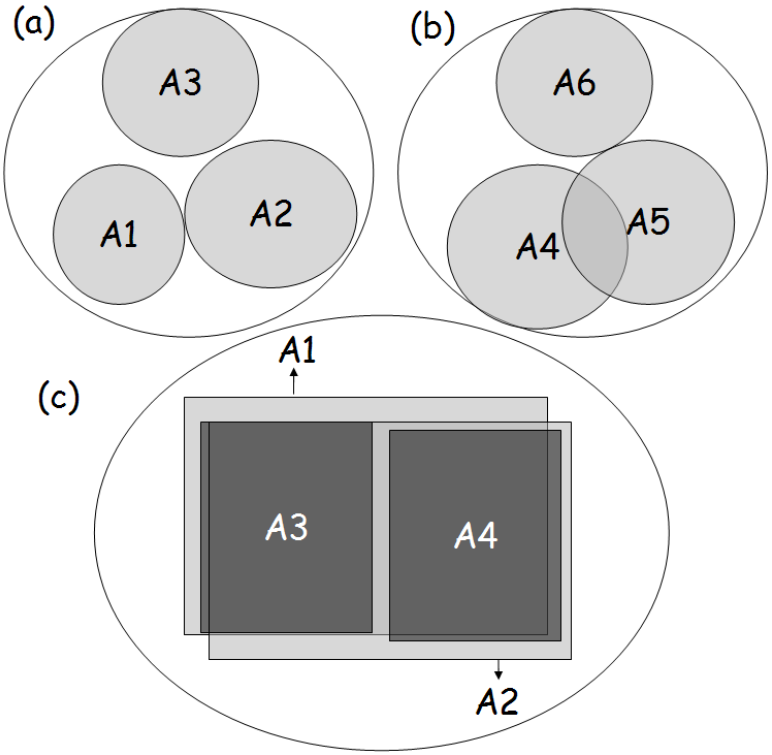

Figure 3.1: Motivating Example

Varying Degrees of Usability - The Genericity-Usability Trade-off: In Figure 3.1(a&b), we assumed that each solution is absolutely useful or not at all useful in solving a prob-

lem. For structured cases such as travel plans, the usability may be qualified by a degree of usability - for example, a route from A to B that includes a long detour is less useful than a direct route from A to B. Even in case of manual usability assessments, users may rate the solution at various degrees of usability. For textual problem-solution repositories, the degree of usability to a particular problem may be assessed automatically by means of statistical text similarity measures when labeled data in the form of *highly usable* solutions is available. In the absence of extrinsic knowledge sources such as ontologies, the usability of highly usable solutions in the labeled data may be propagated to solutions that are lexically similar to it. Figure 3.1(c) illustrates a case of such varying usabilities of solutions; the degree of darkness of the coverage area is directly related to the usability of the solution to that problem subset. As shown in the figure, the coverage areas of A1 and A2 be roughly the same (approx. 60% of the problem space) and that their darkness represents *medium* usability; on the other hand, A3 and A4 cover two disjoint 30% regions with *high* usability. In some sense, A1 and A2, that can solve more problems, are more generic than A3 and A4. Consider the problem of choosing one solution from among $\{A1, A2, A3, A4\}$. We could choose one from either of $\{A1, A2\}$ or $\{A3, A4\}$; any choice from the former set leads to preserving a *medium* usability solution for roughly 60% of the problems, and a choice from the latter preserves a *high* usability solution for half as many. Clearly, it is not the case that one choice is definitely better than the other. The trade-off between these choices, with *generic medium* usability solutions at one end and *specific high* utility solutions at the other, is often tricky, and depends on the intelligence of the downstream adaptation engine (that adapts the retrieved solution at query time in response to a user problem), user preferences etc. For example, a smart user and/or a smart adaptation engine could enable usage of a medium usability solution, whereas less expert users may need a high usability solution to be able to make any use of.

Varying Degrees of Usability - The Role of Abundance: Consider choosing two solutions (instead of just one) in Figure 3.1(c); each combination of two solutions leads to a coverage of 60% of the problem space. One option is to choose A1 and A2 that leads to retaining *two medium usability* solutions for each problem in the 60% space. At the other extreme is choosing A3 and A4 that retains *one high usability* solution for each problem in the same space. This brings in the additional dimension, that of *abundance*, in the trade-off. Whether one would prefer *two medium usability solutions*

or *one high usability solution* is again subjective. The trade-off with the intermediate choices are illustrated in Figure 3.2. The choice in the trade-off is dependent on various factors that relate to the usage of the system. For systems involved in scenarios similar to directory search (to find the contact details of a particular entity of interest), the single high usability solution may be better since it is just one product instance that the user would eventually select. However, in information seeking scenarios and decision support systems such as a *yellow pages* search, the choice providing two solutions for each problem may be desirable since that provides the user with *more* information (and possibly, different perspectives) to make an informed decision. This is related to the query-time problem of diversity-relevance trade-off in information retrieval (Jain *et al.*, 2004).

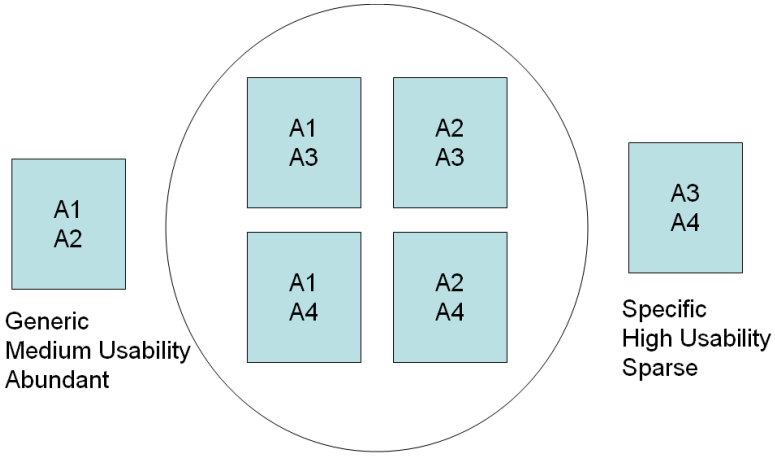

Figure 3.2: Trade-off for Motivating Example

Real scenarios are often more complex than that in Figure 3.1(c); a solution that is of medium usability for a part of the problem space may be of high usability to other problems. Thus, coverage areas for solutions are often composed of various shades. The notion of *usability* is not to be confused with the criterion a knowledge reuse framework would use during retrieval, to estimate whether a particular solution is to be presented in response to a user posed problem. We will outline the differences and provide a crisp definition of *usability* in Section 3.5.

The Trade-offs at the Indexing and Querying Time: Preserving better usability solutions of historical problems would aid providing better usability solutions for unforeseen problems as long as the character of historical and unforeseen problems remain similar - this is a classical assumption in predictive modeling systems, and implicitly applied in CBR by usage of historical repositories in solving unforeseen problems.

For retaining the same number of solutions, preserving generic solutions (that are usable for more problems, on an average) as opposed to specific ones is likely to provide *more* usable solutions per problem. Thus, the compaction time preference for *genericity* translates well to *abundance* of solutions at query time.

3.3 Problem Setting

We now formally introduce the problem setting and the various notations that we will use in later sections. Let the dataset comprising all available problems and solutions be represented as $\mathcal{D}$ and the number of problems it contains be n.

$$\mathcal{D} = [\{p_1, s_1\}, \{p_2, s_2\}, \ldots, \{p_n, s_n\}]$$

Each p_i represents a problem and s_i the set of solutions posed for the problem in p_i, by various users.

$$s_i = \{s_{i1}, s_{i2}, \ldots\} = \{s | s \in Solutions(p_i)\}$$

Our problem of compaction is that of deriving a case base $\mathcal{D}'$ from $\mathcal{D}$ by removal of some solutions from $\mathcal{D}$. Thus, $\mathcal{D}'$ may be represented as:

$$\mathcal{D}' = [\{p_1, s'_1\}, \{p_2, s'_2\}, \ldots, \{p_n, s'_n\}]$$

where

$$\forall i, 1 \leq i \leq n, s'_i \subseteq s_i$$

In compacting $\mathcal{D}$ to form $\mathcal{D}'$, we would like to make solution removal decisions based on a user specified preference of the quality-abundance trade-off.

We do not remove problems during this compaction process since we would not like to reduce the range of problems that can be addressed, in the compaction process. Moreover, given the advancements in Information Retrieval in the past decade, the util-

Notation	Meaning
$p(\alpha)$	The problem associated with the solution α
$s(\rho)$	The solutions associated with the problem ρ
$u(\alpha, \rho)$	This indicates the usability of the solution α for the problem ρ estimated using text similarity metrics when $\rho \neq p(\alpha)$
$u(\alpha, p(\alpha))$	The number of user votes that α has obtained from the voting system employed in the CQA system.

Table 3.1: Notation

ity problem is more manifested in post-retrieval processes that are handled by the CBR system core, making the compactness of the solution set an important priority. With all problems being preserved, each element in $\mathcal{D}$ would have a corresponding element in $\mathcal{D}'$ although there could be elements in $\mathcal{D}'$ with no solutions (i.e., with $s'_i = \phi$). Our compaction process decides on which solutions are to be removed from $\mathcal{D}$ based on a use-specified trade-off between *usability* and *abundance* of solutions; we allow the user to specify the desired trade-off using a parameter γ; a high value for γ is meant to indicate that the usage of $\mathcal{D}'$ would be in scenarios where the user is interested in getting *more* solutions (even at the expense of some reduction in usability), whereas low values indicate strong preference for usability over abundance. We will delve deeper into details of the γ parameter in Section 3.7.

Table 3.1 defines some notation that we will use in later sections. The usability of each solution to its associated problem is intuitively related to the number of votes the solution has received. However, as is obvious, the number of votes is not available in order to assess the usability of a solution to a different problem; in such cases, we estimate the usability using text similarity measures as we will shortly describe.

3.4 Related Work

An overview of the state-of-the-art in maintenance of case bases appears in Section 2.4. As outlined therein, most of the literature for filtering case bases have revolved around maximizing the estimated usefulness of the final compacted version, with respect to a test set of problems. The most common notion of usefulness, called *competence*, as generalized from the concepts outlined in Smyth and Keane (1995), may be outlined as

follows:

$$Competence(C,T) = \bigcup_{c \in C} \{t | t \in T \wedge Adaptable(c,t)\}$$

where C and T denote the compacted case base and the test dataset respectively. The function $Adaptable(c,t)$ determines whether the solution part of c may be used to solve the problem part of t. Such competence models assume that $Adaptable(.,.)$ is a boolean function, and do not account for varying degrees of adaptability (i.e., usability) to the problem. Our problem, on the other hand, is related to exploiting the availability to fine-grained usability estimates along with other information such as number of votes attained by the user, in compacting case bases. Since usability of the solution is available for just the problem for which the solution was posted, we make use of text similarity metrics outlined in Section 2.5.3 to compute the usability of a solution to any problem whenever the solution(s) posted for the problem is/are available.

3.5 Notion of Usability

The notion of usability of a solution to a problem, and its estimation is central to our work. Usability of a solution for a problem quantifies the ease of adapting the solution for solving the problem. A CBR system, when posed with a problem, searches for solutions from among solutions of similar problems in the case base; those solutions are *deemed to be usable* for the posed query and are then presented to the user. However, such solutions that are deemed to be usable may eventually be unusable, since the CBR conjecture may not always hold. Consider the following case:

- **Problem:** How can one travel from Glasgow to Edinburgh in Scotland?
- **Solution:** ScotRail

and a problem, *How can one travel from Keith to Duffington in Scotland?*. Due to the similarity between this new problem and the one in the case base, the solution *ScotRail* would have a high *deemed usability* according to a CBR engine, for the latter problem. However, it turns out that while most towns in Scotland are connected by ScotRail services, Keith and Duffington are connected by a heritage line *Keith Duffington*

Rail that maintains a separate ticketing infrastructure. Thus, the actual usability of the solution *ScotRail* is very close to zero, despite a high *deemed usability*. In the absence of availability of the solution for the latter problem, i.e., *Keith Duffington Rail*, giving *ScotRail* a high deemed usability for the Keith-Duffington problem is clean since it falls out so from the CBR hypothesis of similar problems having similar solutions. It may be noted that the above example is a particularly hostile case where the CBR hypothesis does not hold, and only serves to illustrate the point than being representative of the general case.

The actual usability, however, can be estimated directly for two problems when the solutions for both are available. User voting would aid the computation in our example since most users who try *ScotRail* to book tickets from Keith to Duffington based on the CBR suggestion, would obviously be disappointed and would not vote in favor of the *ScotRail* solution (or may vote *thumbs-down* on it, if the voting system has such an option). If the real solution to the query, (i.e., *Keith Duffington Rail*) is available, the actual usability of the *ScotRail* solution can be estimated as being very low using statistical measures since *ScotRail* shares not even one common word with *Keith Duffington Rail*. It is this *actual usability* of a solution to a problem, a supervised concept - since that can be estimated only with user feedback, or with the availability of true solutions of the problem - that we will refer to by the word *usability*, in the following sections. The difference between this and the notion of deemed usability (that may be estimated even when only the problem is available), we hope, is apparent from the ongoing discussion. In typical cases, unlike in the above example, deemed usability is expected to reasonably approximate supervised usability.

3.6 Computing Usability for Textual Cases

As seen in the previous section, the usability of a *solution* to a *problem* is best determined by user feedback systems such as voting. Alternatively, hints such as user clicking on a link within a solution can be used as indicators of usableness on the lines of relevance feedback (Kelly and Teevan, 2003) in Information Retrieval. However, while dealing with a static corpus of CQA data, the only available user feedback would be votes that indicate the usability of a solution α to the problem for which it was posed,

i.e., $p(\alpha)$. Such votes, adequately discounted for phenomena like vote spam (Agichtein *et al.*, 2008), could be used as a proxy of the usability $u(\alpha, p(\alpha))$. In our compaction process, we often need to estimate the usability of a solution α, to a problem other than the one to which it is associated, i.e., $u(\alpha, \rho)$ where $\rho \neq p(\alpha)$. In such cases, due to the unavailability of votes indicating the relevance of α to ρ, we have to fall back to using statistical measures to estimate the usability. In our static corpus, we have historical solutions collected for every problem that we would need to deal with; we will now outline our estimation of usability of a solution α to any problem ρ in $\mathcal{D}$, with the aid of solutions in $s(\rho)$:

$$u'(\alpha, \rho) = \begin{cases} \#votes(\alpha) & if\ \rho = p(\alpha) \\ max\{u(\beta, p(\beta)) \times sim(\alpha, \beta) | \beta \in s(\rho)\} & else \end{cases} \tag{3.1}$$

where $sim(\alpha, \beta)$ is a text similarity function such as tf-idf cosine (Ref. Section 2.5.3) and $\#votes(\alpha)$ denotes the number of user votes for the solution α. We use $u'(.,.)$ to denote the usability instead of $u(.,.)$ since this definition will be revised very shortly. The usability estimate above is based on the intuition that α is more usable for ρ if it is *highly* similar to *highly* popular solutions associated with ρ. At a boundary case, if α is lexically identical to a solution for ρ with score 5, it is natural to expect the usability $u(\alpha, \rho)$ to have a score of 5 regardless of other lower scored solutions available in $s(\rho)$; the *max* operator ensures this. Cosine similarity measures lexical similarity and could underestimate the semantic match. For example, the strings *"follow subsequent instructions to complete the signup"* and *"directions therein would guide you to register"* have only one word in common leading to a low lexical similarity whereas they are semantically very similar. Such underestimation may be preferred to overestimation (overestimation is expected to be rarer in the case of lexical match based measures) since presenting no solutions is preferable to presenting a lot of un-usable solutions, to the user.

Usage of a CBR system starts with receiving a user problem and invoking a retrieval engine to retrieve the top-k similar problems. Downstream processes look towards processing *only* the set of solutions associated with the top-k similar problems. Thus, an solution α is useful to a problem ρ *iff* the problem $p(\alpha)$ is among the top-k similar

solutions to ρ. We incorporate this notion of being retrieval conscious (Smyth and McKenna, 1998) in estimating usability and outline our final formulation of usability as follows:

$$u(\alpha, \rho) = \begin{cases} u'(\alpha, \rho) & if \;\; p(\alpha) \in \textit{top-k}(\rho) \wedge u'(\alpha, \rho) \geq \beta \\ 0 & \text{otherwise} \end{cases}$$

Thus, the usability $u(.,.)$ is the retrieval-aware refinement of the formulation for $u'(.,.)$. We will set all low usabilities, as determined by a threshold β, to 0 as denoted in the first condition above. This thresholding is necessary to prevent solutions of very low usability to be presented to the user. We will set β consistently to 1.0, the usability equivalent to that of one user vote in a voting based system.

3.7 The Trade-off Parameter

As outlined in Section 3.3, our technique will take a user parameter γ into account, to guide the compaction process. The γ parameter is a way for the user to inform the system regarding the point of preference in the trade-off illustrated in Figure 3.2. At one extreme of the trade-off, the users of the target CBR system are interested in getting the *maximally usable solution* to their queries; at the other, they are interested in getting *as many usable solutions as possible*.

Techniques to cater to such requirements should be able to switch gradually from the usability extreme to the abundance extreme. It is easy to imagine a parameter that varies between 0 and 1, either ends representing either extremes. A system using a dataset compacted according to the *abundance* extreme may present hundreds of *just usable* solutions to each user query; the law of diminishing marginal utility[6], however, suggests that abundance is often not useful beyond a certain tipping point. This limit could either be due to user's haste and/or inability to assimilate large number of results (e.g., as in web search engines where most users do not go beyond two pages of results (Spink *et al.*, 2002)) or the limitations of the adaptation engine (many textual CBR adaptation engines handle just a handful of results (Adeyanju *et al.*, 2010)). *Thus, we feel it is*

[6]http://www.investopedia.com/terms/l/lawofdiminishingutility.asp accessed February 5^{th}, 2014

intuitive to model the trade-off parameter as the tipping point, the number of results upto which a user may consider solution abundance useful.

For example, a user may suggest that she is interested in looking at *upto* 3 different solutions, beyond which she attaches no value to abundance; the compaction engine could then focus on getting enough abundance (possibly, by compromising for usability) that ensures close to 3 solutions for problems from the problem space, beyond which it could optimize for usability. When the parameter is set to 4, the end-user could then expect to receive more number of solutions (closer to 4); however, the trade-off forces it to deliver solutions of lesser usability than the former case with 3. Compaction for certain systems such as technical support and opinion seeking could use a higher value of the trade-off parameter γ, whereas those where the users are expected to be keen on selecting only a particular solution (e.g., directory search) would use a lower value. The parameter γ, given its semantics, thus varies from 1, upwards.

Even if the user specifies a high value of γ, the dataset inherently may not be rich enough to provide γ solutions per problem; at one extreme, the dataset could comprise mostly of highly specific solutions. Consider a dataset that, even in its raw version, is able to provide only 3 solutions for each problem. Since compaction relies on removal of solutions, any compacted version of this dataset also cannot go beyond providing 3 solutions per problem, even if the γ parameter was set by the user to 5. Since it is easy to measure this *physical limit* (i.e., the value of 3 for the example) in a leave-one-out style analysis on the raw dataset, the user may be allowed to set γ only in the range from 1 to the estimated limit.

3.8 Evaluation Measures

We first propose various evaluation criteria to measure the goodness of compaction in our scenario, where the usabilities (of a solution to a problem) are non-binary. Evaluation measures such as competence rely on binary values for usabilities; however, under our setting, solutions presented in response to a problem may have varying degrees of usabilities. The evaluation measures we propose below vary in the method used to aggregate the varying usabilities of the solutions presented for a problem. These are run-time characteristics, that are measured for queries from the test set posed to a

CBR system built on the compacted dataset. The evaluation is done on a test dataset of cases for which true solutions are known (much like classification models are evaluated based on accuracy on a *labeled* test set); the criteria that we describe here all make use of usability scores that are estimated using the true solutions of the test query (these true solutions would have popularity scores such as the number of votes they fetched). Much like in the evaluation of predictive models, the list of solutions for each problem in the test set is estimated by the CBR engine without making use of the solutions available; i.e., the solutions for problems in the test set are used only in estimating the quality of the solutions delivered by the CBR engine (which, by itself, is unaware of the solutions). It may be noted that the evaluating the goodness of a CBR system over a set of test problems is different from evaluating its alignment (Raghunandan *et al.*, 2008); the latter evaluation does not need a test set and attempts to quantify how well the CBR hypothesis holds in the case base. Consider a set of test queries, $\mathcal{T} = \{t_1, t_2, \ldots, t_k\}$. Let the set of solutions from the CBR system for a test query t_i be denoted by $R(t_i)$. The usability of each solution in $R(t_i)$ can be estimated using our notion of usability and the individual solution usability scores may be aggregated to a single score for $R(t_i)$ using any suitable aggregation function (we will outline some functions shortly). For comparing compaction techniques, we will use the average of these measures across problems in $\mathcal{T}$, for each technique.

Usability of the Maximally Usable Solution*(max):* For a time-strapped user or an adaptation engine with extremely limited capabilities, the usability of the maximally usable solution may be the only measure of interest.

$$max(t_i, R(t_i)) = max\{u(\alpha, t_i) | \alpha \in R(t_i) \wedge u(\alpha, t_i) \geq \beta\}$$

where $u(.,.)$ is estimated as described earlier. We will only consider those solutions with usability beyond β as seen in Section 3.6. Note that this max is different from the max set operator (that denotes the numerically largest value in the set) that was used in the definition of $u(.,.)$.

Number of Usable Solutions Presented*(#sol):* In many scenarios, the user may be interested in getting multiple usable solutions for his query; this would enable her to make an informed choice or provide the adaptation engine more options to combine

solutions. Typical scenarios include technical support where there could be multiple legitimate ways of solving a user issues, and information seeking scenarios (e.g., seeking multiple modes of transport in a route-planning problem).

$$\#sol(t_i, R(t_i)) = |\{\alpha | \alpha \in R(t_i) \wedge u(\alpha, t_i) \geq \beta\}|$$

Total Usability of Solutions*(tot):* When solutions are graded on their usability, a user may prefer the $\{high, medium\}$ combination than $\{high, low\}$ even if both fare equally well on the above quality metrics; this is because the total usability of the former is higher than that of the latter.

$$tot(t_i, R(t_i)) = \sum_{\alpha \in R(t_i) \ where \ u(\alpha, t_i) \geq \beta} u(\alpha, t_i)$$

Coverage*(cov):* Consider two test problems and two compacted datasets, one dataset that would provide a large number of very high quality solutions for one problem and none for the other and another that gives a moderate number of medium usability solutions for both. These would fare equally on all the measures so far; however, we would prefer the latter due to a less skewed distribution between test problems. Coverage measures the percentage of problems for which at least one solution has been found; as is evident, coverage is measured over the entire test set $\mathcal{T}$ (unlike the *max #sol* and *tot* metrics above).

$$cov(\mathcal{T}) = \frac{|\{t_i | \#sol(t_i, R(t_i)) \geq 1\}|}{|\mathcal{T}|} * 100$$

cov precisely denotes the notion of *competence* of case bases as generalized from the concepts presented in Smyth and Keane (1995), the state of the art measure in evaluating case base compaction techniques in literature. We call this metric as *coverage* since we think that as a more intuitive terminology to describe what it denotes.

Diversity*(div):* Consider two sets of solutions for a given query that are identical on *max, #sol* and *tot* measures; one would prefer a set that has more diversity that the other (Clarke *et al.*, 2008). At one extreme, the presence of two identical solutions in a set only adds as much value as the presence of either of them. We estimate the diversity

as:

$$div(t_i, R(t_i)) = \frac{\sum_{\alpha_x \in R(t_i), \alpha_y \in R(t_i), x \neq y} (1 - sim(\alpha_x, \alpha_y))}{|R(t_i)|}$$

Since diversity measures the heterogeneity of the solution set and is not defined for a singleton set, $div(t_i, R(t_i))$ evaluates to 0.0 when $|R(t_i)| = 1$.

3.9 Our Approach

We now describe how we condense the input dataset $\mathcal{D}$ to a smaller dataset $\mathcal{D}'$ (the properties of the transformation is as described in Section 3.3; only solutions are removed). We define a measure of fit (an objective function) that is determined using a *leave one out* style evaluation over the dataset, and in each iteration remove the solution that least affects the objective function value. We will see that our design of the objective function is such that it monotonically decreases with removal of solutions; thus, in each iteration, we select to remove that solution that minimally reduces it. We continue these iterations until enough solutions have been removed.

Consider an intermediate dataset $\mathcal{D}^\circ$ ($\mathcal{D}^\circ$ is to be understood as $\mathcal{D}$ after removal of some solutions) and the input compaction parameter γ. Since the specification of the compaction parameter suggests that the user is interested only in the top-γ solutions for a query, our measure of fit will also concern with only the top-γ solutions. Consider a case $\{p_i^\circ, s_i^\circ\}$ (used to denote the case corresponding to the i^{th} case in $\mathcal{D}$) from $\mathcal{D}^\circ$ whose problem p_i° is posed to a CBR system built on $\mathcal{D}^\circ - \{p_i^\circ, s_i^\circ\}$ (much like a classification model is applied to a held out training object in cross-validation style). Our objective function for this specific p_i° is defined as the following:

$$\mathcal{O}(\mathcal{D}^\circ, p_i^\circ) = max(p_i^\circ, R'(p_i^\circ)) + tot(p_i^\circ, R'(p_i^\circ))$$

where $R'(p_i^\circ)$ denotes the top-γ solutions to p_i° when posed on a CBR system that works on the dataset $\mathcal{D}^\circ - \{p_i^\circ, s_i^\circ\}$. We use the original solution set s_i (from $\mathcal{D}$, since it is a superset of s_i° and hence more richer) to estimate the $u(.,.)$ for usage in computation of $max(.,.)$ and $tot(.,.)$ above.

Alg. 1 ***Greedy Decremental Optimization***

Input. $\mathcal{D}, \kappa, \gamma$
Output. $\mathcal{D}'$

1. $\mathcal{D}^\circ = \mathcal{D}$
2. while $\mathcal{D}^\circ$ has more than κ solutions
3. $\quad a = \underset{\alpha \in \bigcup_i s_i^\circ}{\operatorname{argmin}} (\mathcal{O}(\mathcal{D}^\circ) - \mathcal{O}(\mathcal{D}^\circ \backslash \alpha))$
4. $\quad$ remove a from among the solutions in $\mathcal{D}^\circ$
5. return $\mathcal{D}^\circ$ as $\mathcal{D}'$

The design of $\mathcal{O}(\mathcal{D}^\circ, p_i^\circ)$ counts the usability of the maximally usable solution twice, once in the first term and once in the second term; the other solutions are only counted once (in the second term, as part of the $tot(.,.)$). For very high γ, the overall sum is expected to be contributed to, by mostly the second term since $|R'(.)|$ increases with γ and $tot(., R'(.))$ increases with the former. Thus, this construction of $\mathcal{O}(\mathcal{D}^\circ, p_i^\circ)$ reduces the relative contribution of the maximally usable solution for high γ (as is intuitively desired, since high γ implies that the user is interested in getting many solutions).

The overall measure of fit for $\mathcal{D}^\circ$ is then defined as:

$$\mathcal{O}(\mathcal{D}^\circ) = \sum_{p \in \{p_i^\circ | \{p_i^\circ, s_i^\circ\} \in \mathcal{D}^\circ\}} \mathcal{O}(\mathcal{D}^\circ, p)$$

Consider that we want to retain only upto κ solutions in the compacted dataset $\mathcal{D}'$; it is now easy to formulate a greedy decremental approach (that starts with $\mathcal{D}$ and discards solutions based on those that reduce the $\mathcal{O}(.)$ minimally) that we present in in Algorithm 1 (GDO).

Although the objective function only includes $max(.,.)$ and $tot(.,.)$, Algorithm 1 indirectly optimizes on coverage. This is because the removal of the only usable solution to a query (i.e., the operation that would reduce coverage) would affect the objective function through both the $max(.,.)$ and $tot(.,.)$ (whereas most other operations only affect one of these) and hence is less preferred by design. This is so since the only solution is also the maximally usable solution.

3.9.1 Complexity Analysis

Here we briefly analyze the complexity of GDO. Each iteration involves finding the closest problems to each problem in $\mathcal{D}^\circ$. Since the problems in $\mathcal{D}^\circ$ remain the same across iterations, we pre-compute and store the top similar problems to each query at a worst-case complexity of $O(|\mathcal{D}|^2)$; however, state of the art IR techniques are much faster. Within each iteration, for each problem in $\mathcal{D}^\circ$, we would have to analyze the usability of the solutions associated with each of the top-q similar problems (q is typically a constant in a CBR system). The usability computation involves comparing solution sets pair wise at a complexity of $O(r^2)$ where r is used to denote the average size of a set of solutions. Since this has to be done for each problem ($|\mathcal{D}|$ of them) against its top-q similar problems, the complexity of the operations within each iteration evaluates to $O(|\mathcal{D}|qr^2)$. Let the number of solutions to be removed (i.e., the number of iterations) be t; this leads to an overall complexity of $O(|\mathcal{D}|qr^2t)$. The only parameter that GDO is quadratic in, i.e., the average number of solutions per problem, is found to be less than half-a-dozen on the average.

3.9.2 Baseline Techniques

Making use of heuristically estimated and graded solution usabilities in textual case base compaction is a new problem, and has not been addressed in literature. All filtering techniques proposed for case base maintenance so far assume usability to be binary (i.e., a solution is either usable for a problem or not). *Thus, we can only compare our techniques with those that work with binary usabilities arrived at by quantizing the estimated usability with a threshold.* This comparison helps us to illustrate that considering usabilities from a continuous domain for case base compaction, leads to better compacted datasets. The latest algorithm for case base mining (Pan *et al.*, 2007) makes use of *solution categories* to perform slightly better than the case deletion policy (DEL) from (Smyth and Keane, 1995) and the case addition policy (ADD) from (Zhu and Yang, 1999); both DEL and ADD are generic techniques that can work without solution categories. The unavailability of solution categories in textual datasets (that we use for our experiments) forces us to fall back on the DEL and ADD techniques to compare our technique with. However, we need certain adaptations to be able to

incorporate γ that we outline here.

ADD: This algorithm (Zhu and Yang, 1999) works by starting with an empty case base (in our case, an empty solution set), choosing solutions to add based on the number of new problems that each solution could cover. Thus, in contrast to our setting where $\mathcal{D}^\circ$ starts as $\mathcal{D}$ and evolves to a smaller set, ADD starts with a $\mathcal{D}^\circ$ comprising of only the problems and a gradual enrichment with solutions leads it to the final compacted version. In our scenario, even if a problem has been covered by a pre-chosen solution, it is useful to having another solution to the query since the user is deemed to be attaching value to upto γ solutions per query. Consider choosing from among two solutions, one of which provides a solution to a query that has no pre-chosen solution, and the other that provides a solution to a query that already has γ-1 solutions; we would intuitively prefer the former. Accordingly, we now outline a function that quantifies the benefit of a candidate solution α being considered for addition to $\mathcal{D}^\circ$ as $bft(\alpha, \mathcal{D}^\circ, \gamma)$:

$$bft(\alpha, \mathcal{D}^\circ, \gamma) = \sum_{p_i^\circ} \begin{cases} 1.0, & \text{if } u(\alpha, p_i^\circ) \geq \beta \wedge \#S(p_i^\circ, \mathcal{D}^\circ) = 0 \\ 0.5, & \text{if } u(\alpha, p_i^\circ) \geq \beta \wedge \#S(p_i^\circ, \mathcal{D}^\circ) < \gamma \\ 0.0, & \text{otherwise} \end{cases}$$

where $\#S(q, \mathcal{D}^\circ)$ denotes the number of usable solutions for q already present in $\mathcal{D}^\circ$. This formulation attaches a higher value to choosing a first solution, and a lower value for subsequent ones. Once a query has already accumulated γ solutions, the benefit of adding a solution to that query is made to drop to 0. An algorithm that greedily chooses solutions that maximize the $bft(.,.)$ to include (at each step) is easy to visualize; we omit the pseudo code hence.

DEL: It is non-trivial to adapt the ideas of auxiliary, pivot, support and spanning cases in the decremental approach (Smyth and Keane, 1995) to include the γ parameter. However, we present a best-effort approach here. We define *auxiliary* solutions as those whose deletion does not affect the competence, i.e., each of the queries for which it is useful already have at least γ other solutions. Then, we delete solutions in the order of those that have least benefit according to the $bft(.,.)$ function on the current dataset as outlined above. This leads to a simple decremental approach.

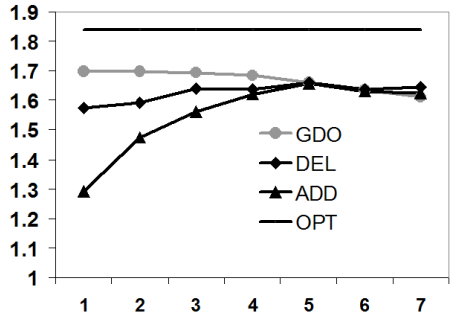

Figure 3.3: Experimental Results: *max* vs. γ

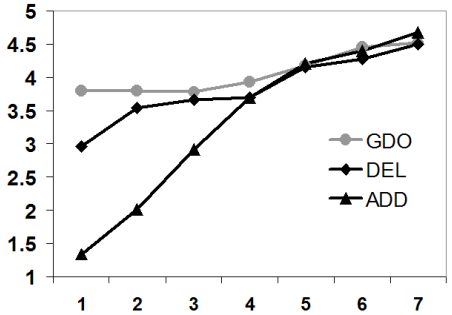

Figure 3.4: Experimental Results: *#sol* vs. γ

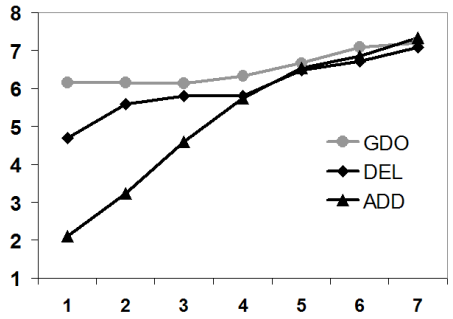

Figure 3.5: Experimental Results: *tot* vs. γ

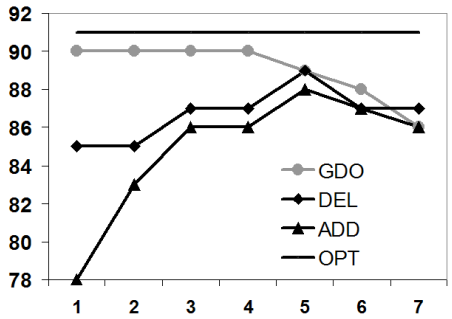

Figure 3.6: Experimental Results: *cov* vs. γ

3.10 Experimental Evaluation

We now experimentally evaluate our GDO technique against the ADD and DEL approaches. Having laid down the algorithms and the evaluation criteria, the design of the experiments is fairly obvious. We evaluate the algorithms on each of the 5 evaluation criteria described in Section 3.8 on varying compaction efforts (i.e., varying κ in Algorithm 1) and varying values of γ. We describe the datasets used in our experiments and our empirical analysis by varying γ and compaction efforts in the following sections.

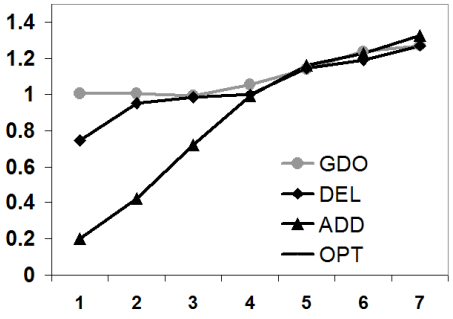

Figure 3.7: Experimental Results: *div* vs. γ

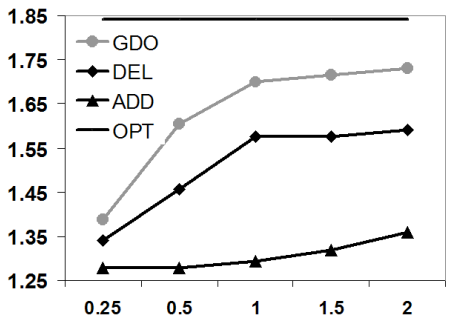

Figure 3.8: Experimental Results: *max* vs. ratio (encarta) (γ=1)

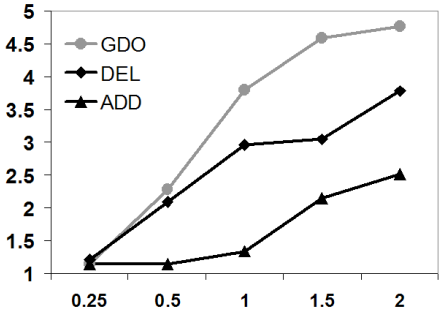

Figure 3.9: Experimental Results: *#sol* vs. ratio (encarta) (γ=1)

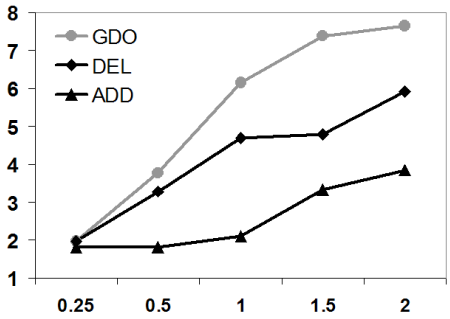

Figure 3.10: Experimental Results: *tot* vs. ratio (encarta) (γ=1)

3.10.1 Datasets

Among the datasets described in Section 2.7, the CQA datasets are most relevant to our problem since our problem is motivated by the need for filtering CQA datasets. Compaction by removal of solutions is most relevant to scenarios where there are multiple solutions per problem; *encarta* is the only non-CQA dataset among those in Section 2.7 that has multiple solutions per problem. Based on such considerations, we use three datasets for our experimental study; (1) MSR Question Answering Dataset[7] (*encarta*), (2) Yahoo! Answers Question-Answers from the *Lincoln cars* category (*lincoln*) and (3) Yahoo! Answers Question-Answers from the *Delhi and NCR* category (*delhi*). Each of these have multiple scored solutions per problem. We randomly chose 100 problems and their solutions to use as a test set. We use the remaining as input to the compaction algorithms, $\mathcal{D}$, and collect the compacted dataset, $\mathcal{D}'$. For each problem in the test set, we choose solutions from the top-k similar problems to it (we set k to 3, in our experiments) from $\mathcal{D}'$ and evaluate that set on the various measured in Section 3.8; an aggregate of such measures across problems in the test set is reported herein. The number of problems and solutions in each of the datasets, before compaction, are provided in Table 3.2. To avoid monotony, we present charts for the results on *encarta* dataset, and provide details about the results on other datasets when there are differences in the trends that are worth mentioning. Though our experiments show consistent trends across the various datasets on the random test/train split, experiments with larger datasets and different test/train splits would help assess the generalizability of our results better.

[7] http://research.microsoft.com/en-us/downloads/88c0021c-328a-4148-a158-a42d7331c6cf/default.aspx accessed February 5^{th}, 2014

Dataset	#Problems in $\mathcal{D}$	#Solutions in $\mathcal{D}$	#TestProblems	#TestSolutions
encarta	1026	9025	100	814
lincoln	1303	6132	100	459
delhi	1434	7414	100	526

Table 3.2: Dataset Statistics

3.10.2 Performance with Varying γ

In our empirical analyses, we start by analyzing the performance of the various techniques with varying γ. *We set κ, the number of solutions to be preserved in $\mathcal{D}'$, to be the number of problems in $\mathcal{D}$ ensuring one solution per problem in $\mathcal{D}'$, on an average*; this, obviously, leads to different compaction efforts per dataset since they have different $\frac{\#solutions}{\#problems}$ ratios to start with. γ is directly related to the desirability for more solutions. The performance on the *max* measure is expected to fall with increasing γ, since *more* and *better* are contradictory goals. We plot the performance of the various techniques on the *encarta* dataset in Figures 3.3,3.4,3.5,3.6 and 3.7 (γ is plotted on the X-Axis). The OPT line (wherever it is visible on the scale) shows the best possible value of the metric (which is achieved when no compaction has been performed); this signifies the upper bound.

From the charts, GDO is seen to outperform the other techniques on each and every measure for small values of γ, while the difference between the various techniques diminish at high values of γ. GDO fares 9%, 30%, 30% and 6% better than the next best technique on the *max*, *#sol*, *tot* and *cov* measures respectively at $\gamma = 1$. The average number of solutions per query in the *lincoln* and *delhi* datasets were significantly smaller to start with (Ref. Table 3.2); thus, the compaction effort is smaller, leading to lesser differentiation in the measures on them. The GDO technique remained the best performer on each of the settings on the other datasets; we summarize the quantum of improvements in Table 3.3. It may be noted that exploiting usability scores in compaction is seen to drive better performance on the $\#sol$ and div measures as compared to the baseline techniques that target optimizing on $\#sol$ (due to usage of binary usabilities). This highlights the importance of graded usabilities in that it automatically leads to $\mathcal{D}'$ containing more and diverse solutions for test set problems.

Dataset	Compaction Effort	max	#sol	tot	cov	div
encarta	8.93 times	9%	30%	30%	6%	34%
lincoln	4.69 times	3%	15%	17%	1%	28%
delhi	5.14 times	2%	8%	7%	3%	14%

Table 3.3: Results with γ=1 (%age GDO is better than 2^{nd} best)

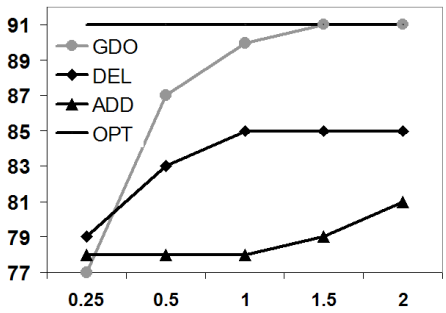

Figure 3.11: Experimental Results: *cov* vs. ratio (encarta) (γ=1)

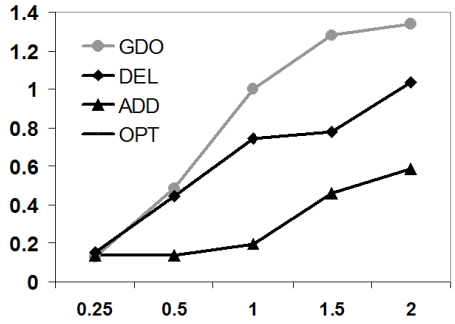

Figure 3.12: Experimental Results: *div* vs. ratio (encarta) (γ=1)

3.10.3 Performance with Varying Compactions

We now illustrate the relative performance of the compaction techniques with varying ratios of $\frac{\#solutions}{\#problems}$ in the compacted dataset, $\mathcal{D}'$. We empirically evaluate the techniques on two settings, one on a dataset compacted with γ=1 and the other with γ=3. The charts for the *encarta* dataset with γ=1 are plotted against the $\frac{\#solutions}{\#problems}$ ratio (in $\mathcal{D}'$) in Figures 3.8,3.9,3.10, 3.11 and 3.12 (ratio plotted in the X-axis). As is expected, at extremely low ratios (e.g., 0.25), only very few solutions get retained, leading to the techniques converging at very low values for the various measures of interest. The difference in effectiveness between the techniques manifest more as more solutions can be chosen for retention. Much like in experiments in Section 3.10.2, GDO mostly outperforms the baseline techniques in every measure of interest, under varying ratios (i.e., compaction efforts). Similar observations were found for γ=3 as well; charts are presented in Figures 3.13,3.14,3.15,3.17 and 3.16. The margins between the techniques are narrower when compared against those for γ=1. This is expected since the techniques are found to converge with increasing γ (Ref. Section 3.10.2). Similar trends were observed for the *lincoln* and *delhi* datasets.

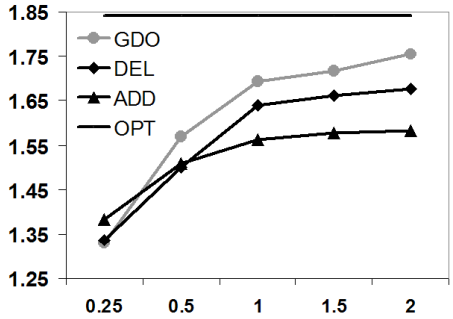

Figure 3.13: Experimental Results: *max* vs. ratio (encarta) (γ=3)

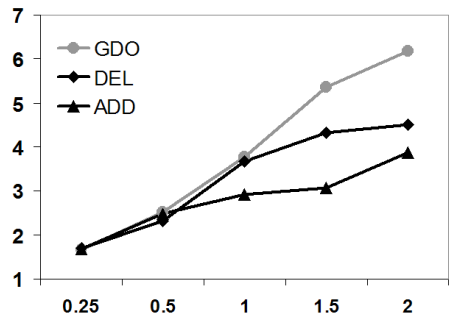

Figure 3.14: Experimental Results: *#sol* vs. ratio (encarta) (γ=3)

3.11 Discussion

We have seen that GDO outperforms ADD and DEL on all the performance measures outlined in Section 3.8 on most settings that we have experimented with, with varying margins. However, some of these observations were expected; for example, GDO directly optimizes on a sum of the max and tot measures on the training data and hence may be expected to perform better than ADD and DEL on these measures intuitively. In this section, we discuss the significance of observations from our experiments, separately for each of the evaluation measures that we consider.

max and tot: The $\mathcal{O}(.)$ function used by GDO is modeled as to take the max and tot into account over the dataset that is compacted. On the other hand, since ADD and DEL deal only with crisp usabilities, they do not consider the max and tot factors that quantify usabilities. This intuitively leads to an expected advantage for the GDO approach, that reflects in the results.

#sol: The $\mathcal{O}(.)$ function optimizes for the sum of usabilities of the γ solutions, and thus optimizes for $\#sol$ conditional on the user input, γ. The $bft(.,.)$ function utilized by both ADD and DEL also have a similar component, since they count the number of answers available in a weighted fashion (1.0 for the first answer, and 0.5 for upto the remaining $\gamma - 1$ answers) upto γ answers. Despite this similar nature with respect to $\#sol$, the GDO approach is found to significantly outperform the others on the $\#sol$ criterion. This suggests that using fine-grained usability estimates (than binary estimates) in preferring solutions to retain, across problems, automatically leads to retention of solutions that are usable across many problems. This inference stems from the fact that GDO outperforms the other approaches although all the three approaches optimize on

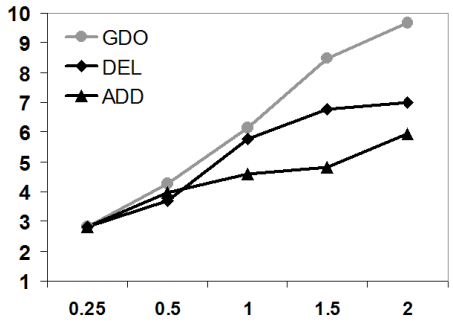

Figure 3.15: Experimental Results: *tot* vs. ratio (encarta) (γ=3)

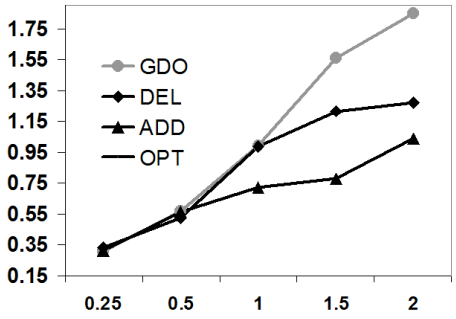

Figure 3.16: Experimental Results: *div* vs. ratio (encarta) (γ=3)

$\#sol$; the only extra thing that GDO does is to consider the usability estimates without thresholding to binary values, and thus the improved performance may be attributed to it.

cov: The observations on cov is more like that for $\#sol$ above; though all of GDO (since it discourages removal of the only solution to a problem more than removing one of many as observed in Section 3.9), ADD and DEL (the $bft(.,.)$ function attaches a higher weight to the only solution) vie to retain at least one solution for as many problems as possible, GDO is seen to outperform the others. The margins on cov, however, are not seen to be as large as those for $\#sol$; yet, the observations on cov also enable us to assert that considering fine-grained usability estimates helps in preserving solutions that cover more problems, though the improvements aren't very high.

div: This measures the diversity of usable solutions; higher diversity is desired since that is likely to provide orthogonal information such as diverse viewpoints or independent solutions. However, none of the $\mathcal{O}(.,.)$ or $bft(.,.)$ functions have been designed to optimize for this criterion. Nevertheless, GDO is seen to outperform ADD and DEL significantly (see, for example, Figure 3.16) on this measure. This may also be attributed again to the differences in exploiting usability estimates, since that is the prime factor that differentiates GDO from ADD and DEL. Thus, highly usable solutions across problems may be inferred to be inherently diverse too; thus, considering usabilities automatically optimizes for diversity, to an extent. This is partly due to users typically tending to vote only for the earlier or already popular solution if multiple solutions are very similar or identical; whether such behavior is prevelant in CQA systems needs further investigation.

It is fairly obvious from the analysis of the observations that exploiting fine-grained usabilities in compacting QA datasets is a good idea for many reasons. Using statistical measures of usability estimates derived from voting systems in compaction is seen to help in building better CBR systems for unforeseen problems (such as those in the test set that we evaluate on); specifically, it helps in ensuring more solutions per problem ($\#sol$), betters chances of finding at least one solution for a problem (cov) and in providing more diverse solutions (div).

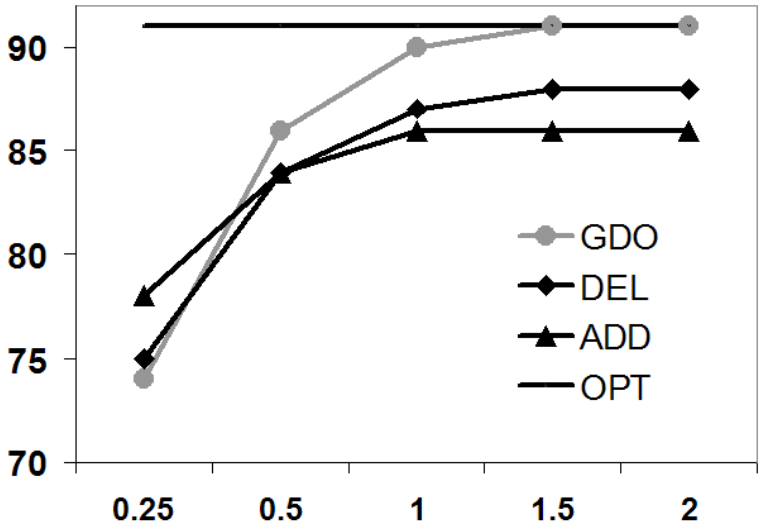

Figure 3.17: Experimental Results: *cov* vs. ratio (encarta) (γ=3)

3.12 Chapter Summary

In this chapter, we considered the novel problem of harnessing text-similarity based measures in estimating the usability of solutions, and exploiting such measures in compacting problem solution datasets. We outlined the specific challenges involved in utilizing such usability estimates; most importantly, the trade-off between usability and abundance of solutions. In our context of textual problems and solutions, we outlined means to make use of popular text similarity measures and user feedback data to arrive at intuitive estimates of usability. We introduced several quality measures to evaluate the goodness of the compacted dataset for usage in a CBR system. We then presented a greedy optimization technique, GDO, that exploits statistical estimates of usabilities in compaction. Using an extensive set of experiments, we demonstrated the effectiveness of our approach (by comparing against best-effort based adaptations of existing techniques) on multiple datasets and evaluation measures. Most importantly, heeding

to usability scores is seen to improve on the number of usable solutions that could be provided for each problem.

The usability scores that we currently use rely on statistical measures and underestimate the actual usability. Using tools such as wordnet to derive a semantic usability score would improve the effectiveness of techniques such as GDO that work by exploiting such scores.

CHAPTER 4

SEGMENTATION OF TWO-PART TEXT DOCUMENTS

4.1 Introduction

Various kinds of text data that are available from sources ranging from personal web pages and enterprises contain problem-solution information. Travel experience narrations often contain reusable information such as travel tips (e.g., *"do not respond to locals offering help near the railway station since their service would most likely be a rip off"*) whereas medical diagnosis reports have a list of symptoms followed by the diagnosis. Thus, documents that contain problem-solution information have varying degrees of separation between the problem and solution parts with some such as travel warnings being almost of the kind where the problem and solution parts are intermingled as shown above. As observed in Section 2.3.2, separating out problem solution information is hard since it involves various sub problems as the following:

1. Modeling of problem and solution behavior to identify characterizations for the two kinds of text.

2. Using such modeling to extract problem and solution parts from text documents.

3. Mapping the problem and solution parts thus extracted to create problem-solution pairs to be used as cases in a case base.

In this chapter, we consider segmentation of two-part text documents as a first step towards the larger goal of extracting cases from text documents. We use two-part text documents to refer to any document that can be interpreted as comprising of two parts. A typical example of a two-part text document is that of incident reports which describe various events relating to a problem and its solution in chronological ordering. An incident report would typically start with a set of statements describing the problem and

Bug Reports *(Software Development)*	Problem: Bug Description Solution: Resolution
Medical Transcription	Problem: Symptoms Solution: Diagnosis and Drug Prescriptions
Incident Reports	Problem: Incident Description Solution: Solution Steps to Resolve the issue
RCA (Root Cause Analysis) Reports *(Service Delivery)*	Problem: Problem Ticket Description Solution: Solution Applied
Legal Case Reports	Problem: Crime or Offence Solution: Outcome of the Case (e.g., Verdict)

Table 4.1: Two-part Text Documents in Various Domains

how it was observed, followed by textual narrations of the solution steps used in solving the problem. An example incident report from an airline company is given below:

A Nexen employee checked his rebreather unit and found it to be past its "Service Due Date". He alerted the other passengers and they found the same situation with their re-breather units. All the units were promptly changed by the heliport staff and the flight proceeded as normal.

Of the above text, the first two sentences describe the nature of the incident whereas the third indicates how the problem was solved. With incident reports mostly being written with a chronological description of events, the problem part would mostly precede the solution part in such documents. Even if no chronological ordering is explicitly enforced in other domains such as medical diagnosis and judicial procedures, the part analogous to the problem part (symptoms, or crime description, as the case may be) would intuitively almost always precede the "solution" part i.e., the diagnosis or the verdict respectively; this is so since the problem part provides enough context that is a pre-requisite to assimilate the solution part. We list some domains in which we have encountered two-part text documents along with the most appropriate problem and solution segment mappings therein, in Table 4.1.

Another aspect that is worth noticing in such two-part text documents is that a lot of such similar documents (e.g., from the same domain) are often available as a collection. For example, one could procure all bug reports for a specific software and they would all have similar linguistic behavior in terms of the vocabulary and style. This opens

up the possibility of building collection-level models out of such domain-specific collections which could then be further used to segment each document individually into the problem and solution parts. The advantage of using such collection level models as opposed to document-level models is that the former would intuitively be more noise-robust due to having been learnt over a collection. We will use such considerations in developing a technique for segmenting two-part text documents, in this chapter.

4.2 Related Work and Background

Among the state-of-the-art techniques that could address the problem of separating out problem solution parts from two part text documents, are text segmentation techniques. As outlined in Section 2.3.2, text segmentation techniques leverage linguistic differences to split text documents into component segments. The linguistic features that are harnessed by text segmentation algorithms include word repetitions (Reynar, 1994) and co-reference across candidate segment boundaries (Passonneau and Litman, 1997).

The problem of segmenting two-part text documents, however, poses a hard problem for such text segmentation algorithms. This is because of the lexical-interrelatedness between the two segments, due to them referring to related incidents. For example, the term *rebreather unit* has similar chances of occurring in the problem and solution units in the incident reports pertaining to it. The ongoing discussion is not to say that there is no difference in character between problem and solution segments, but, that they could be fewer in our scenario as compared to traditional scenarios where text segmentation techniques are deployed (such as segmenting a text book into chapters, or any text document where a coherent text segment is related to a *subtopic*). However, we will compare the technique we develop against the state-of-the-art text segmentation algorithm that works on one document at a time, i.e., APS (Kazantseva and Szpakowicz, 2011).

4.2.1 Segmentation using Collection-level Models

While most segmentation algorithms segment one document at a time, our scenario of the availability of multiple related documents leads us to exploring ways of exploiting

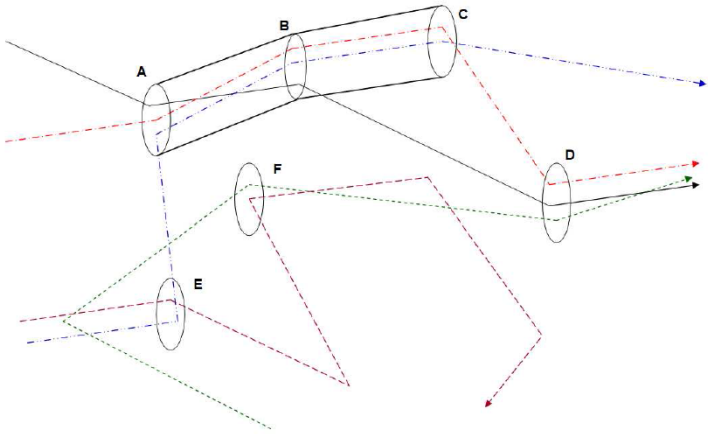

Figure 4.1: Example of Collection-level Segmentation in CBA; Figure from (Kummamuru *et al.*, 2009)

such collections in segmenting documents in the collection. The only work we have come across, that learns and uses collection-level models for text segmentation relates to the domain of segmentation of call transcripts from a contact center (Kummamuru *et al.*, 2008, 2009). It represents each transcript as a sequence of turns (i.e., sentences) and uses a four-step approach to segment transcripts in an unsupervised fashion. Consider sentences to be represented as points in a multi-dimensional space (each sentence represented by the point corresponding to the tf-idf vector); then, each call transcript may be represented as a line connecting the multi-dimensional points in the sequence in which they appear in the transcript. Such a visualization appears in Figure 4.1, and we will use the example to illustrate the four-phase approach as below:

- *Sentence Clustering:* Sentences across various documents are clustered using lexical similarity measures to produce *sub-step* clusters. Such sub-step clusters comprise lexically coherent sets of sentences, as represented by the ovals in Figure 4.1.

- *Ordering-Similarity based Clustering:* Such sub-steps are then clustered using ordering similarity in documents. For example, if two sub-steps mostly contain sentences from the same part (e.g., beginning, middle, end) of documents, they are more likely to be clustered together. Such clusters of sub-step clusters are called *representative segments*. The tunnel covering three ovals (i.e., sub-steps) in the Figure 4.1 is an example of such a representative segment.

- *Segmentation:* Each document is then represented as a sequence of representative segment IDs based by replacing each sentence by the representative segment it belongs to. This leads to an intuitive segmentation where contiguous sequences having the same ID may be thought of as segments. In our example, the three sentences in the red call that fall within the tunnel would be assigned to the same segment; so is the case with the two sentences in the black call that are together within the tunnel.

- *De-noising:* The segmentation arrived at from the previous step, could be very fragmented. For example, each sentence could be assigned a different segment ID; two large sequences with the same ID may be separated by a single sentence that has a different ID. A variant of hierarchical agglomerative clustering is then used to get rid of such noisy segment assignments so that documents are ultimately segmented into large contiguous segments.

We will refer to this **C**lustering-**B**ased **A**pproach as CBA, and will use it in our experimental study as a baseline method to compare against.

4.2.2 Affinity Propagation for Segmentation

APS (Kazantseva and Szpakowicz, 2011) is one of the latest text segmentation algorithms from literature. Each basic unit, i.e., a sentence, is treated as a data point; similarities between sentences are estimated using lexical measures, or any other means that are available. The central idea in APS is inspired from an earlier affinity propagation algorithm for clustering where each cluster is associated with an exemplar (which is one of the data points itself) that is meant to represent the cluster. The iterative process involves passing two kinds of messages:

- *Availability($i \rightarrow j$):* The weight of this message is directly related to the likelihood that the sender i is the examplar for the receiver j, given the evidence from all other data points.

- *Responsibility($i \rightarrow j$):* Analogous to the availability message, this indicates the likelihood that the receiver j is the exemplar for the sender i, given the evidence

from all other potential exemplars.

Once the estimates converge, the iterative process is stopped, and the clustering as determined by the final estimates is output. Since segmentation requires that sentences belonging tothe same segment need to be adjacent (and not scattered all over the document), an adjacency constraint is included in the APS formulation. As is the case with most text segmentation algorithms, this approach could generate one or many segments, and need not necessarily discover two segments, as we need in our case. We will compare the technique we develop against APS too, during our experimental evaluation.

4.2.3 IBM Model 1 Translation Model

To solve the text segmentation problem for two-part text documents, we need to address challenges due to the lexical inter-relatedness of segments, as outlined in Section 4.1. Towards modeling lexical inter-relatedness, we will use statistical machine translation models. In this section, we provide some background into the simplest and earliest statistical machine translation model, IBM Model 1 (Brown *et al.*, 1993). Machine translation deals with the problem of converting text data from one language to another. The machine translation process involves a training phase to learn a mathematical model which is then applied on documents in the source language to produce translations.

- **Training:** This phase requires a large (the larger, the better) corpus of parallel corpus for the language pair that we are interested in modeling. Parallel corpus is used to refer to a set of documents that have a two-part structure, one being a section written in the source language (say, English), followed by its translation in the target language (say, French). An example (mini) document from such a corpus could read [*This is my home, C'est ma maison*]. Relatedness between words is discovered using the parallel corpus and captured in a mathematical model. In IBM Model 1, the mathematical model may simply be represented as a set of $p(f|e)$ probabilities that are learnt for every word french-english word pair f, e.

- **Using the translation model:** Given the model and a document in the source language, the translated version of the document (in the target language) can readily

be obtained. To illustrate the idea, one could simply replace each word, e in the source document by the most likely word $translated(e)$:

$$translate(e) = \arg\max_{f} \quad p(f|e)$$

where f denotes any word in the target language, and $p(f|e)$ denotes the probability learnt using the translation model. It need to be asserted that this is a very simplistic illustration of the machine translation process intended to convey the high-level idea of machine translation, and the real process is fairly intricate.

The training process that generates the model probabilities internally employs an expectation maximization algorithm. Though the translation model was proposed to translate documents from one language to the other where the vocabularies of the source and target language are disjoint, it has been widely used for a variety of purposes. The one closest to our scenario is that of enhancing retrieval in problem-solution repositories (Xue *et al.*, 2008); a translation model is built using the problem-solution document as an element in the parallel corpus whereby the training process would learn probabilities such as:

$$p(w_i \in solution|w_j \in problem)$$

that indicate the propensity of the problem-solution document authors to choose w_i for inclusion in the solution when they have chosen w_j for inclusion in the problem. Retrieval on problem solution repositories may be enhanced (Xue *et al.*, 2008) using such models by implicitly *"expanding"* the query to include solution words (since the query would most likely contain problem-like words, but has to be matched against documents that contain both problems and solutions); the detailed retrieval scoring function appears in Section 2.5.5. IBM Model 1, the statistical machine translation model that we will use in our approach, is the earliest and simplest machine translation model; over the last two decades, several improved models have been proposed in literature, to remedy the shortcomings of the basic model. However, given that efforts in usage of translation models in retrieval such as Xue *et al.* (2008) have stuck with IBM Model 1, we also attempt to leverage the same in our formulation. Additionally, many of the

improvements over the basic model (e.g., fertility, re-ordering) have been motivated by issues around translating text from one natural language to another; such issues do not necessarily manifest in our intent to simply capture word correlations across problems and solutions.

Notation: Translation models may be thought of as 2-d associative arrays. The translation model $\mathcal{T}$ would maintain the probability of v occurring in the solution when w occurs in the problem at $\mathcal{T}[w][v]$. For consistency with notation in previous literature, we will denote that probability term as $\mathcal{T}(v|w)$. $\mathcal{T}(.|w)$ then refers to the multinomial distribution of solution words indicating the probability of occurrence whenever w occurs in the problem.

4.2.4 Language Models

Among the tools that we will use in our segmentation technique are language models. Language models (Song and Croft, 1999) model the lexical behavior of a set of documents as a set of word probabilities conditional on the occurrence of zero or more words. Unigram language models are the simplest among language models and assume that words occur independent of each other. Thus, they simply learn one probability term per word. We provide a brief toy example to illustrate unigram language models without getting into details such as maximum likelihood estimation and smoothing. Consider a sentence S as *"unigram language models learn unigram probabilities"* and a unigram language model built over this sentence. For every word occurring in the sentence, we associate a probability as:

$$p_S(w) = \frac{freq(w, S)}{\sum_{w' \in S} freq(w', S)}$$

where $freq(w, S)$ counts the frequency of the word w in S. Accordingly, we would estimate $p(\text{"}unigram\text{"})$ at 0.33 since it occurs twice (thus, 2 in the numerator) and associate 0.167 with all other words. We can then estimate the probability of generating a sentence (or phrase) S' using this language model as:

$$P_S(S') = \prod_{w \in S'} p(w)$$

Thus, the probability of the phrases *unigram probabilities* and *language probabilities* would be 0.055 and 0.028 respectively. This is intuitive since the first phrase contains the word *unigram* that occurs more frequently in the source sentence than *language* (which occurs as the corresponding word in the second phrase). Thus, a unigram language model is a probability distribution over words occurring in the corpus over which it was learnt.

4.3 Problem Setting

We now formally define the problem of segmentation of two-part text documents. Given a set $\mathcal{C}$ of n two-part text documents of a similar nature, denoted as,

$$\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$$

where each document C_i comprises of l_i words,

$$C_i = [w_{i1}, w_{i2}, \ldots, w_{il_i}]$$

we would like to identify a segmentation vector $\mathcal{Z}$, of n values $\{z_1, z_2, \ldots, z_n\}$, which is used to represent the segmentation of every document C_i as:

$$Problem(C_i) = [w_{i1}, w_{i2}, \ldots, w_{iz_i}]$$

$$Solution(C_i) = [w_{i\ (z_i+1)}, \ldots, w_{il_i}]$$

We would like $\mathcal{Z}$ to approximate as closely as possible the actual segmentation of each document C_i into its two inherent problem and solution segments. Though the segments may have to be called differently - e.g., as symptoms-diagnosis segments for the medical domain, problem-resolution segments for a collection of bug reports - we will consistently use the same terminology and refer to them generically as problem and solution segments.

For document C_i, we will limit our search for values of z_i to only those values that

will partition C_i at the sentence boundaries. Thus, we will ensure that sentences are not broken up; i.e., with a segment of a sentence being put in the problem part and the remaining in the solution part. dom_{C_i} will be used to denote all possible sentence boundaries in the document C_i.

4.4 Correlation and Cohesion driven Segmentation (CCS)

We will now describe our technique for segmentation of two-part text documents that we will call as *Correlation and Cohesion driven Segmentation*, abbreviated to CCS. Text segmentation techniques work using the assumption of intra-segment lexical cohesion and thus discover segments that are lexically coherent. However, in addition to such assumptions, we will use some additional ones as alluded to in the discussion in Section 4.1. We start by listing those assumptions in more detail, and then delve into the details of the CCS approach.

4.4.1 Assumptions used in CCS

Our approach relies on four major assumptions regarding behavior of the collection of two-part text documents.

Separation: The two parts of each document, viz., the problem and solution parts, are assumed to be well-separated. In particular, we expect that sentences from the problem part and those from the solution part do not appear intermingled in the document flow. The worst case would be a document where alternate segments belong to the problem part separated by sentences from the solution part, whereas the best case would be all the problem and solution sentences separately bundled together.

Segment Ordering: Most text segmentation algorithms segment the text document into multiple segments without attaching a meaning/semantics to each segment. Algorithms that need to use a collection of documents, such as CBA, correlate segments across documents since the models themselves are learnt at the collection level. Our approach will also follow a similar strategy as CBA, and thus needs to be able to reason across common segments (i.e., problem and solution segments) in various documents. In addition, we would also like to label each segment as either problem or solution

rather than simply segmenting a document into two parts. To overcome such issues, we assume that the relative ordering of segments across documents follows the *problem followed by solution* pattern. This is not very restrictive as observed in Section 4.1 since the problem often sets the context and hence needs to come prior to the solution.

Similar Problems have Similar Solutions: The assumption that similar problems tend to have similar solutions is at the core of Case-based Reasoning. This is the underlying principle behind selecting solutions of similar problems to suggest as potential solutions for a new problem, in the usage of a CBR system (Aamodt and Plaza, 1994). At the word level, this assumption asserts that there ought to be correlations between some words in the problem parts and some words in the solution parts. Translation models are an intuitive tool to learn such correlations that could then be used in the segmentation phase.

Intra-segment Lexical Cohesion: This is a classical assumption used in text segmentation algorithms starting from TextTiling (Hearst, 1994). This assumes that the lexical dispersion of words within a segment would be less; segmentation algorithms score boundaries where the lexical similarity among sentences on the same side is large, as likely segmentation boundaries. Since we identify segments as either problem or solution, we extend the assumption to say that the set of sentences that belong to the problem (solution) segment across documents are likely to be lexically coherent.

Despite listing down the assumptions prominently prior to presenting the technique, we do not necessarily need these assumptions to be *absolutely* adhered to, to do meaningful segmentations. As is with the case of any statistical/learning technique, we only need these assumption to hold *mostly*. Minor deviations from such assumptions - e.g., 3-4 documents having solution followed by problems in a corpus of 1000 documents, are easily tolerated - as long as the deviations are not statistically large enough to largely influence the models that we will learn, to do the segmentation.

4.4.2 A Generative Model for Two-part Text Documents

We will now outline a generative model to model the authoring of two-part text documents. Let us assume that $\mathcal{P}$ is a unigram language model learnt over the problem segments of segmented two-part text documents, and that $\mathcal{S}$ is an analogous solution

language model. The translation model generated by IBM Model 1 when fed with such segmented two-part text documents is denoted by $\mathcal{T}$. Given the models $\{\mathcal{P}, \mathcal{S}, \mathcal{T}\}$, we model the generation of a document C_i having the first z_i words in the problem and the remaining $(l_i - z_i)$ words in the solution as being in accordance with the following generative process:

1. For each position i from 1 to z_i,
   (a) Sample word w_{ij} from the problem unigram model $\mathcal{P}$
2. For each position i from z_{i+1} to l_i,
   (a) Choose a random number between 0 and 1 as r
   (b) If r is less than ψ, */* with a probability of ψ */*
      i. Sample w_{ij} from the solution unigram model $\mathcal{S}$
   (c) else */* with a probability of $(1 - \psi)$ */*
      i. Sample w_{ij} from the average of the multinomial models in the set $\{\mathcal{T}(.|w_{i1}), \ldots, \mathcal{T}(.|w_{z_i})\}$

Informally, our generative model assumes that the document author would choose words from the problem language model to create the text in the problem part. The words in the solution parts are either generic solution words sampled from the solution language model, or problem-specific solution words sampled from the average of the $\mathcal{T}(.|w)$ distributions where w is any word already chosen for the problem part. Further, we use ψ as a parameter to indicate the relative preference between the two sources of solution words; the solution words are assumed to be sampled from the solution language model with a probability of ψ and from the translation model source with a probability of $(1 - \psi)$ as indicated in the pseudo code above.

Illustrative Example: We now illustrate the rationale behind the generative model described above using an illustrative example. Consider a toy two-part document from a IT helpdesk-like scenario, with the following structure:

Problem: Disk full reported

Solution: Some files were deleted to resolve the issue

Among the various solution words *{files,deleted,resolve,issue}* (discarding stopwords *were* and *to*), it is intuitively likely that *resolve* and *issue* are among the words that the solution language model is highly skewed in favor of, given that these appear to be words that are likely to appear with high frequency across all IT helpdesk solutions. On the contrary, *files* and *deleted* are likely to occur with high frequency among solutions to a disk problem, and would hence be well supported by the translation model conditioned on words such as *disk* and *full* that appear in the problem part of the document. We illustrate this intuition in Figure 4.2. Our generative model allows for dual origin of solution words to account for such intuitive characteristics of solution words.

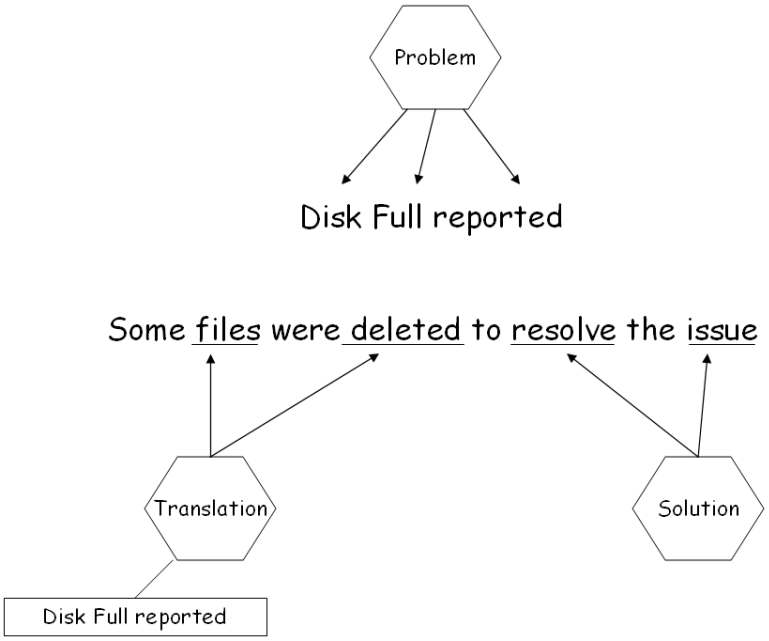

Figure 4.2: Illustrative Example for Generative Model

Probability Computation: Given that we have outlined the generative model, we can use it for computing the probability of generating a document C_i (segmented into two parts at z_i) using the models in $\{\mathcal{P}, \mathcal{S}, \mathcal{T}\}$ as follows:

$$p(C_i, z_i|\mathcal{P}, \mathcal{S}, \mathcal{T}) = \prod_{1 \leq j \leq z_i} \mathcal{P}(w_{ij}) \times \prod_{z_i < j \leq l_i} (\psi\, \mathcal{S}(w_{ij}) + (1-\psi)\, avg\{\mathcal{T}(w_{ij}|w_{i1}), \ldots, \mathcal{T}(w_{ij}|w_{iz_i})\})$$

The above construction for probability computation falls out from the generative model. The product of the problem word probabilities according to the problem lan-

guage model is computed for the first z_i words since they are sampled directly from the problem language model. For each solution word, we compute the weighted sum of probabilities from the solution language model and the averaged translation model conditioned over the problem words; the weights are set to ψ and $(1-\psi)$, the probabilities of using these sources of solution words as assumed in the generative process. The word probabilities are then aggregated in a product form to derive the probability of generating the document under the generative process using the models in $\{\mathcal{P}, \mathcal{S}, \mathcal{T}\}$.

We can now aggregate the probabilities of the individual document segmentations across documents to estimate the probability of the $\mathcal{C}, \mathcal{Z}$ combination, given the models $\{\mathcal{P}, \mathcal{S}, \mathcal{T}\}$:

$$L(\mathcal{P}, \mathcal{S}, \mathcal{T}; \mathcal{C}, \mathcal{Z}) = p(\mathcal{C}, \mathcal{Z}|\mathcal{P}, \mathcal{S}, \mathcal{T}) = \prod_{C_i \in \mathcal{C}} p(C_i, z_i|\mathcal{P}, \mathcal{S}, \mathcal{T})$$

where $L(\ldots)$ denotes the likelihood function. We will use θ as a shorthand for $\{\mathcal{P}, \mathcal{S}, \mathcal{T}\}$ for notational convenience.

4.4.3 Iterative Process and the Objective Function

For the probability computation, we assumed the availability of good language and translation models; given such models, we can then choose the best values of z_i that are supported by the models. However, the models themselves can be estimated well only if the segmentation points are available. This cyclic nature leads to a straightforward iterative process that proceeds in the following manner, within each iteration:

- **Segmentation:** Given the current estimates of the models θ, we can learn the best segmentation points for each document as those that score the highest according to the probability computation as estimated from the models.
- **Model Learning:** With the current estimate of the segmentation vector $\mathcal{Z}$, we can then (re-)learn the models. This is easy since the segmentation points split the documents into problem and solution parts; the language models can be learnt over the respective segments, and the translation model could be learnt using the parallel corpus of problem-solution documents.

The above 2-part iterative step is motivated by the expectation and maximization steps of the classical EM algorithm (Dempster *et al.*, 1977). We will now outline the objective function, and describe the EM algorithm according to the above framework, that seeks to optimize the objective function.

For the set of documents $\mathcal{C}$, we need to estimate the models that are most likely to generate the document set. The objective function that needs to be maximized, hence, is the likelihood of the data given the models, represented as:

$$L(\mathcal{P}, \mathcal{S}, \mathcal{T}; \mathcal{C}) = L(\theta; \mathcal{C}) = p(\mathcal{C}|\theta)$$

Our probability computation from Section 4.4.2 can be applied only when the segmentation vector $\mathcal{Z}$ is known. In its absence, we can compute the likelihood estimate by marginalizing over all possible values of $\mathcal{Z}$:

$$p(\mathcal{C}|\theta) = \sum_{\mathcal{Z}' \in domain(\mathcal{Z})} p(\mathcal{C}, \mathcal{Z}|\theta)$$

$$= \prod_{C_i \in \mathcal{C}} \sum_{z \in domain(z_i)} p(C_i, z|\theta)$$

We will outline the EM approach in the next section.

4.4.4 The EM Approach

To optimize $p(\mathcal{C}|\theta)$, there are two sets of unknowns, the models (i.e., θ) and the segmentation vector $\mathcal{Z}$. In the E-step, we will use the current estimates of the models to compute the posterior probabilities of each of the segmentation points. The M-step then comprises of re-estimating the models in accordance with the posterior probabilities of the various values from the various dom_{C_i}s.

E-Step: According to the generative model from Section 4.4.2 and the objective function in Section 4.4.3, the maximum likelihood estimates of the models may be obtained if we know the following:

- The correct segmentation point z_i for every document C_i.

- Information for each solution word in C_i as to whether it was derived from the solution language model or the translation model.

In the E-step, we estimate the above unknown information based on the current estimates of θ. In particular, we perform the following steps:

1. For each document $C_i \in \mathcal{C}$,
   (a) For each segmentation point $z' \in dom_{C_i}$
   (b) */* i.e., for each segmentation point in document C_i */*,
      i. Estimate $p(z'|C_i, \theta)$, the probability that z' is the correct segmentation point for C_i based on the current θ.
      ii. For every solution word w in C_i,
         A. Estimate $p(Source = \mathcal{S}|w, C_i, z', \theta)$, the probability that w was derived from the solution language model.
         B. Estimate $p(Source = \mathcal{T}|w, C_i, z', \theta)$, the probability that w was derived from the translation model.

Thus, the E-step may be summarized by the equations to find the three probability terms in the above flow. The first probability term $p(z'|C_i, \theta)$ is easily estimated by conditioning the distribution $p(C_i, z'|\theta)$ from Section 4.4.2 over the document C_i as below:

$$p(z'|C_i, \theta) = \frac{p(C_i, z'|\theta)}{\sum_{z \in dom_{C_i}} p(C_i, z|\theta)}$$

For the source estimation, the probability of a word being generated from the solution language model is directly related to the probability of the word in the multinomial distribution of $\mathcal{S}$. Similarly, the source being the translation model is directly related to the probability of the word in the averaged conditional translation model as shown in Step 2.c.i in the generative process described in Section 4.4.2. We also need to account for the prior probabilities of choosing either of these sources, as indicated by ψ in generative model. Additionally, given that a word needs to come from either of these

sources, the source probabilities should sum up to 1.0. The following formulae ensure this:

$$p(Source = \mathcal{S}|w, C_i, z', \theta) = \frac{\psi \times \mathcal{S}(w)}{\psi \times \mathcal{S}(w) + (1.0 - \psi) \times avg\{\mathcal{T}(w|p)|p \in problem(C_i, z')\}}$$

$$p(Source = \mathcal{T}|w, C_i, z', \theta) = \frac{(1.0 - \psi) \times avg\{\mathcal{T}(w|p)|p \in problem(C_i, z')\}}{\psi \times \mathcal{S}(w) + (1.0 - \psi) \times avg\{\mathcal{T}(w|p)|p \in problem(C_i, z')\}}$$

In each of the above equations, the denominator sums up the probability estimate of w coming from $\mathcal{S}$ (the term before the $+$ mark) and the estimate corresponding to $\mathcal{T}$; scaling it by this sum ensures that the source probabilities sum up to 1.0 as desired.

M-Step: The M-Step uses the posterior probabilities estimated in the E-step to re-learn the language and translation models. We will use an example-based approach to illustrate the M-step.

We start with some examples of simple unigram language modeling over weighted documents. Consider two documents C_1 and C_2 which are represented as follows:

$$C_1 = \{one = 1.0, small = 1.0, doc = 1.0\}, wgt(C_1) = 0.5$$

$$C_2 = \{large = 1.0, doc = 1.0\}, wgt(C_2) = 0.7$$

The first document C_1 contains three words, each with unit frequencies; the whole document itself has a weight of 0.5. Though fractional word possibilities are not practical in real documents, we may assign words fractional frequencies due to various reasons, the simplest being tf-idf weighting. Now, the language model learnt over such weighted documents would assign the following weight to a word w':

$$\mathcal{L}(w') = \frac{\sum_{C_i} wgt(C_i) \times freq(C_i, w')}{\sum_{w''} \sum_{C_i} wgt(C_i) \times freq(C_i, w'')}$$

where $freq(C_i, w')$ represents the frequency of w' in C_i. For the language model learnt over C_1 and C_2 above, the probability of word *"doc"* would then be:

$$\mathcal{L}(doc) = \frac{(0.5 \times 1.0) + (0.7 \times 1.0)}{(0.5 \times 1.0) + (0.5 \times 1.0) + (0.5 \times 1.0) + (0.7 \times 1.0) + (0.7 \times 1.0)}$$

The terms in the denominator correspond to the words *one, small, doc (from C_1), large* and *doc (from C_2)*. Consider the document C_1 that has two possible segmentation points (we will ignore the sentence boundary restriction for segmentation points, for the sake of illustration), one that splits the document after *one* and the other that splits it after *small*. Let us further assume that the segmentation points have been estimated to have posterior probabilities of 0.3 and 0.7 respectively. Then, C_1 would contribute the following documents to the problem language model:

$$\{one = 1.0\}, wgt = 0.3$$

$$\{one = 1.0, small = 1.0\}, wgt = 0.7$$

If some weighting scheme were used to calculate word weights in the document vector (instead of signifying presence by 1.0 and absence by 0.0), the corresponding word weight needs to be replaced for 1.0 above. Such adaptations need to be applied in the remaining formulae in this section too. To generalize, the document each C_i would contribute to the problem language model for each segmentation point z' is denoted by:

$$C_p(C_i, z') = [\{w = 1.0 | w \in Prob(C_i, z')\}, wgt = p(z'|C_i, \theta)]$$

where $Prob(C_i, z')$ denotes the set of words in the problem part of C_i when split according to the segmentation point z'. The solution language model contributions by each C_i, z' pair is slightly more complex, since the words would need to be weighted according to their source probabilities associated with the solution language model. This is so since the solution language model should not highly weight a word if it is to be accounted for, by the translation model. Thus, the fraction document for the solution language model is:

$$C_s(C_i, z') = [\{w = p(Source = \mathcal{S}|w, C_i, z', \theta)|w \in Sol(C_i, z')\}, wgt = p(z'|C_i, \theta)]$$

Having defined the contributions by each C_i, z' pair to each of the language models, learning of the problem and solution language models is straightforward (as illustrated in the toy example in the beginning of this section).

To learn the translation model, we need document pairs, where the problem and solution parts are documents themselves. IBM Model 1 then learns word correlations among a corpus of such document pairs. We now outline the contribution of each C_i, z' pair to the translation model as a document pair $(T_p(C_i, z'), T_s(C_i, z'))$:

$$T_p(C_i, z') = [\{w = 1.0|w \in Prob(C_i, z')]\}, wgt = p(z'|C_i, \theta)]$$

$$T_s(C_i, z') = [\{w = p(Source = \mathcal{T}|w, C_i, z', \theta)|w \in Sol(C_i, z')\}, wgt = p(z'|C_i, \theta)]$$

For the sake of completion, we now outline the various contributions by the toy document C_1 for its second segmentation point z_{12} to all the models in θ. Let us assume that the source probability of the word *doc* to have come from the solution and translation models are assumed to be 0.4 and 0.6 respectively.

$$C_p(C_1, z_{12}) = [\{one = 1.0, small = 1.0\}, wgt = 0.7]$$

$$C_s(C_1, z_{12}) = [\{doc = 0.4\}, wgt = 0.7]$$

$$T_p(C_1, z_{12}) = [\{one = 1.0, small = 1.0\}, wgt = 0.7]$$

$$T_s(C_1, z_{12}) = [\{doc = 0.6\}, wgt = 0.7]$$

Thus, each language model and the translation model would have as many documents to learn over as there are [document, segmentation point] pairs. The translation model is learnt over weighted documents using a straightforward adaptation of the IBM

Alg. 2 ***CCS***

Input. $\mathcal{C}$, a set of documents
Output. $\mathcal{Z}$, a vector denoting the segmentation

1. *Segment documents in* $\mathcal{C}$ *using state-of-the-art techniques to initialize* $\mathcal{Z}$
2. *Estimate the models* $\mathcal{P}$, $\mathcal{S}$ and $\mathcal{T}$ *using* $\mathcal{Z}$
3. *while* $(p(\mathcal{C}|\mathcal{P}, \mathcal{S}, \mathcal{T})$ *has not yet converged)*
4. **E-Step:** *Estimate the segmentation point posterior probabilities and solution word source probabilities*
5. **M-Step:** *Re-estimate the language and translation models using the E-step probabilities*
6. $\forall \mathcal{C}_i \in \mathcal{C}$
$$z_i = \underset{z \in dom_{C_i}}{\operatorname{argmax}}\, p(\mathcal{C}_i, z|\mathcal{P}, \mathcal{S}, \mathcal{T})$$
7. return $\mathcal{Z}$

Model 1 EM algorithm (Brown *et al.*, 1990) to account for fractional document weights.

4.4.5 The CCS Algorithm

We now outline the CCS approach in pseudo code in Algorithm 2. Since the $\mathcal{Z}$ and θ are learnt iteratively, we can start with an initial estimate for $\mathcal{Z}$. Accordingly, we initialize $\mathcal{Z}$ using segmentations derived from a state of the art segmentation algorithm such as APS (Kazantseva and Szpakowicz, 2011). Since text segmentation algorithms could segment a document into more than two segments, we will use a post-processing step to choose the best from among the set of segmentation points output from the algorithm used in step 1. Our post-processing step chooses the best segmentation point according to the TextTiling score (Hearst, 1994). TextTiling scores each candidate segmentation point according to the lexical disparity between sentences on either side; thus, we choose the one segmentation point that scores highest whenever there are multiple segmentation points obtained from the algorithm used for initializing the segmentation.

We use the initialized $\mathcal{Z}$ to kick start the iterative process that involves the alternating E and M steps as described in Section 4.4.4. We let the iterations run until the objective function converges or till 20 steps, whichever is lower. Once the iterations are done, we set the segmentation for each document C_i as that point in dom_{C_i} that maximizes the probability of generating the document according to the final estimates of the models in θ. The segmentation vector thus formed is output as $\mathcal{Z}$. An overview

of the process appears in Figure 4.3.

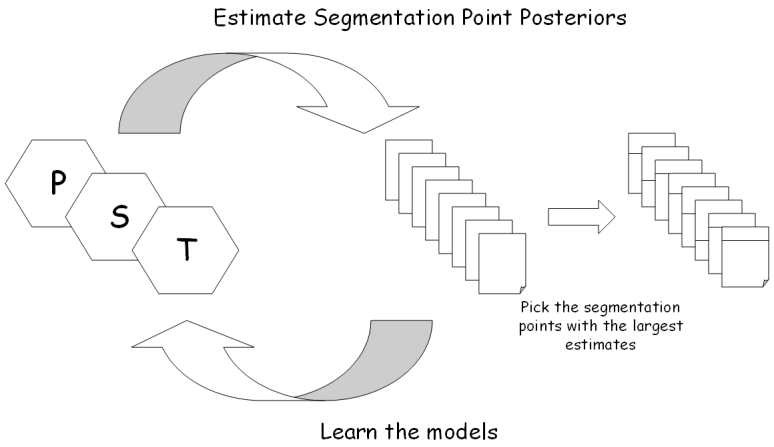

Figure 4.3: CCS Process Overview

CCS has a parameter ψ that determines the relative importance that is given to the solution and translation models. Since we do not have any estimate as to the relative importance of these sources of solution words, we set ψ to 0.5 so that one of the sources is not set to be more preferred by design.

Time Complexity of CCS: Consider a corpus of n documents with a vocabulary of size m where each document has an average of l sentences, each comprising of an average of w words. The E-step computations of posterior probabilities cost $\mathcal{O}(lnw^2)$ whereas the M-step operations of learning the translation and language models cost $\mathcal{O}(k(nw^2 + m))$ and $\mathcal{O}(nw + m)$ respectively when the number of EM iterations in the IBM Model 1 learning is denoted by k. Thus, the total time complexity of CCS when run for k' iterations is of the order of $\mathcal{O}(k'nw^2(k + l) + k'km)$. The only term CCS is quadratic on, is w which is the number of words in a sentence; this is usually bounded and within 7-10 in most document corpora and hence, is not a cause for much concern.

4.5 Experimental Evaluation

In this section, we describe our experimental evaluation wherein we compare the CCS technique against the APS and CBA approaches from literature. We also use TextTiling as a baseline approach , in our experimental analyses. We start by describing the various datasets used in our experimental study, followed by a brief introduction to the segmentation evaluation measures we employ. This is followed by a detailed description of our extensive empirical study which also evaluates the robustness of CCS to noise.

Table 4.2: Datasets and Sizes

Dataset	#Docs	#Sents per doc	#Words per doc
visa	511	4.22	49.75
health	433	5.98	72.61
agri	229	5.10	70.38
loan	514	4.90	61.88
tourism	134	4.79	60.95
railways	268	4.51	50.57
telecom	103	4.70	47.48
web	109	4.08	49.27

4.5.1 Datasets

Since our text segmentation algorithm is intended for segmenting two-part text documents, we choose from among those datasets that have a two-part problem-solution structure; this excludes multi-part (i.e., multi-answer) datasets such as the CQA ones and the Encarta dataset. Memory constraints (since the entire translation model is held in memory) forced us to exclude large datasets such as *career* and *insurance* from our evaluation. That leaves us with the FAQ datasets among those listed in Section 2.7. An overview of the datasets with statistics of word count appear in Figure 4.2. The unavailability of large domain-specific datasets encompassing problem-solution information restricted us to the usage of datasets comprising a few hundreds of documents; however, our experiments show that our approach outperforms the baselines by large and statistically significant margins. Since we employ statistical models that are expected to improve with dataset sizes, we expect that the quality of the CCS segmentation will only improve with the dataset sizes.

These datasets are not of the true two-part text document category; this is so since the source FAQs themselves have the problem-solution separation. Additionally, these have some style differences between the problem and solution parts, due to being derived from FAQ pages. The problem part is often a first person narrative whereas the solution part is more instructive. An example document is illustrated in Table 4.3. CCS does not exploit such style differences explicitly apart from the style differences being implicitly captured in the language models. Such implicit accounting for style differences is done in other segmentation algorithms too; for example, the CBA approach

Table 4.3: Example Problem & Solution from *loan*

My home was appraised by VA and now I am having problems with its condition. Since the appraisal is an inspection of the property, I think the VA should be able to help me with the problems.
Although the VA fee appraiser must view the property from both the exterior and interior to determine its overall condition, the appraisal process is not intended to be an "inspection" of the property. . . .

exploits the similarity of segments (of the same type) across documents and thus can easily capture differences between occurrence frequencies of style-determining words. The intra-segment lexical cohesion criterion in APS would similarly make use of style differences. Thus, we argue that any performance improvement recorded by CCS over and above the state-of-the-art algorithms can be attributed mostly to the CCS formulation.

For each dataset, we will join the problem solution sections to create a single document where the problem appears before the solution. We mask the split-point between segments and employ the segmentation algorithm on them. The segmentation points discovered by those algorithms are the evaluated against the true segmentation point that is available (since we joined the two parts in the first place).

4.5.2 Evaluation Measures

Text segmentation evaluation itself has been a subject of extensive research, and many evaluation measures have been proposed. The most recent one that we know of is the *WindowDiff* measure (Pevzner and Hearst, 2002). In addition to *WindowDiff*, we will use other evaluation measures. We describe each of them herein.

WindowDiff: Consider two segmentations z_1 and z_2 of a document comprising of l sentences. *WindowDiff* (Pevzner and Hearst, 2002) is a penalty score that evaluates to 0 if the two segmentations are identical, with increasing values indicating increasing differences between the segmentations. *WindowDiff* may be thought of as sliding a window of width w across two versions of the document, one segmented according to z_1 and the other segmented according to z_2. At each step, *WindowDiff* counts the number

of segment boundaries within the window for each of the segmentations; whenever the number of segment boundaries are not equal, it increments the penalty score. The formula is as follows:

$$WD(z_1, z_2) = \frac{\sum_{1 \leq i \leq (l-w+1)} equals(\#SB(z_1, i, w), \#SB(z_2, i, w))}{l - w + 1}$$

where $\#SB(z, i, w)$ counts the number of segment boundaries according to z among the w sentences starting from the i^{th} sentence in the document. The $equals(.,.)$ function returns 1.0 when the input arguments are equal, and 0.0 otherwise. w is chosen as half the average segment size.

P_K: P_K (Beeferman *et al.*, 1999) is the precursor to *WindowDiff* and is a very similar penalty measure that uses a sliding window approach. The penalty is added not when the number of segment boundaries are different as it happens in *WindowDiff*, but when there is inconsistency as to whether the two ends of the sliding window are in the same segment among the two segmentations being compared. The same window width is used as *WindowDiff*.

Diff: The above measures for comparing segmentations were proposed to compare general text segmentations where the number of segments is not necessarily two. In our case of two-part segmentation where there is a single segmentation split point per segmentation, a simple metric would be a penalty measure that takes the value of the number of sentences in between the two segmentation points (one from each segmentation). Thus, if z_1 puts 5 sentences in the problem segment and z_2 estimates the problem segment to contain 8 sentences, the *Diff* measure would evaluate to 3. This intuitively measures how many sentences the estimated segmentation is *off* by, and hence, is more understandable to humans than the earlier segmentation quality metrics.

CBR Usability Measures: The motivation behind two-part segmentation is to enable usage of a wider spectrum of textual data for Case-based Reasoning. Thanks to advances in natural language processing and information retrieval, we can estimate the usabilities of a solution to any problem when the solution to the latter is also available. We outlined various metrics for computing such CBR usability in Section 3.8. Among such metrics, we will use the *max* and *tot* metrics in our evaluation. In particular, for a corpus of segmented two-part documents (segmented according to a segmentation al-

Table 4.4: WindowDiff Evaluation

Dataset	TT	CBA	APS	$CCS_{\mathcal{L}}$	CCS
visa	0.307	0.483	0.216	0.158	**0.085**$\star$
health	0.379	0.309	0.302	0.106	**0.030**$\star$
agri	0.329	0.419	0.200	0.135	**0.052**$\star$
loan	0.369	0.402	0.261	0.223	**0.042**$\star$
tourism	0.373	0.398	0.215	0.188	**0.079**$\star$
railways	0.341	0.380	0.213	0.138	**0.015**$\star$
telecom	0.374	0.380	0.276	0.269	**0.186**$\star$
web	0.366	0.480	0.213	0.147	**0.018**$\star$
Average	0.355	0.406	0.237	0.170	**0.063**

gorithm), we pick the problem part of each document; we will evaluate the usability (*max* or *tot* measure, as the case may be) of solutions retrieved by CBR from the other documents, for the chosen problem. This is then averaged across all documents, leading to a single value for each [segmentation algorithm, dataset] pair.

Statistical Significance: For each triplet *[segmentation algorithm,dataset,evaluation metric]*, we obtain a single value indicating the quality of the segmentation of the dataset by the technique over the chosen evaluation metric. Even if a technique fares better than another on such a single measure, we would not have any means of distinguishing between cases such as (1) the better technique is slightly better than the other over *all* documents in the dataset (for the chosen evaluation metric), and (2) the better technique is better by large margins over a small subset of documents, while being largely similar to the other in most documents. Among the above scenarios, the former is preferable since the better technique is consistently better. The *consistency* of a technique in being better than the other is typically measured by statistical significance tests which measure conformance to various p-values. Simplistically, p-values may be thought of as being inversely related to the consistency by which the better technique is better; a p-value of < 0.05 indicates that the chances of the better technique being better by just coincidence (i.e., chance) is less than 5%. We will use randomization tests (Smucker *et al.*, 2007) to assess statistical significance of the performance of CCS with respect to the baseline techniques at a p-value of < 0.05.

4.5.3 Segmentation Quality Evaluation

Evaluation over the WindowDiff Measure: We evaluate the accuracy of segmentation of each technique over each dataset, on each evaluation measure we outlined in Section 4.5.2. In the CCS approach, we use two kinds of statistical models to decide on the segmentation point. The usage of the language model is derived from the assumption of intra-segment lexical cohesion, that is widely used across text segmentation techniques. The usage of translation models is driven by the expectation of correlation of words between problems and solutions, which is unrelated to any assumptions used in classical text segmentation. To explicitly quantify the utility of the translation model, we use a version of CCS that does not use the translation model; this corresponds to setting $\psi = 1.0$ in the CCS approach. We will call that as CCS$_L$ and will include that too, in our evaluation. The relative performance of CCS and CCS$_L$ against the baseline techniques are listed in Table 4.4.

WindowDiff is a penalty measure where the best technique would score least. We represent the best performing baseline technique with an underline in Table 4.4, whereas the best technique overall is represented in bold. It may be seen therein that CCS outperforms all the baseline techniques by large margins across datasets. The best among the baseline techniques, APS, is seen to have upto 10 times larger *WindowDiff* numbers than CCS, thus indicating that CCS would definitely be the most preferred technique for two-part text segmentation. The utility of the translation model is evident from the improved performance of CCS with respect to CCS$_L$, the margins being very high in many cases.

The results of the statistical significance tests are indicated by a $\star$ in Table 4.4; the presence of a $\star$ against CCS indicates that the better performance of CCS is statistically significant over its nearest competitor at a p-value of < 0.05. CCS is seen to provide statistically significant performance improvements on all datasets.

P_K and Diff Evaluation: The trends on the P_K and *Diff* measures were remarkably similar to those observed for the *WindowDiff* evaluation. In particular, CCS was able to bring down the P_K and *Diff* penalty scores by upto 3 times from the values recorded for APS, the best performing baseline technique. The absolute values of P_K and *Diff* for CCS when averaged across datasets were 0.19 and 0.87 respectively. It may be

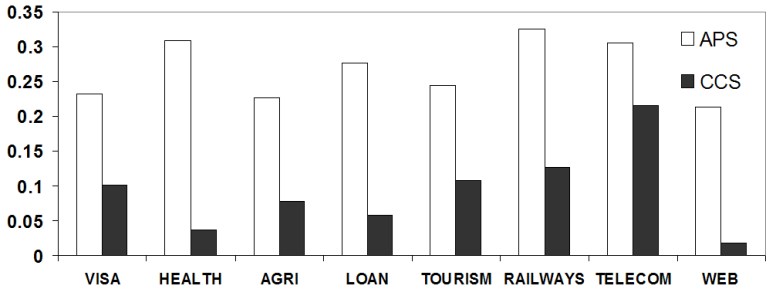

Figure 4.4: APS and CCS on the P_K Measure

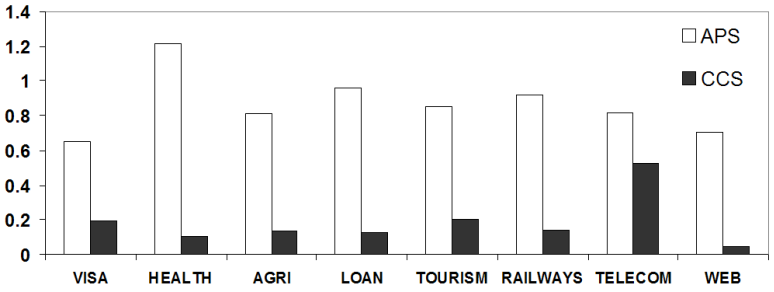

Figure 4.5: APS and CCS on the Diff Measure

noted that 0.87 is a good value for the *Diff* measure since it indicates that the CCS segmentations were less than one sentence away from the true segmentations; *Diff*, unlike *WindowDiff* and P_K does not have 1.0 as an upper bound. For the *Diff* measure, the improvements achieved by CCS over APS were seen to be statistically significant on each dataset under a p-value of < 0.05. The comparative performance of CCS and APS over the P_K and *Diff* measures are plotted in Figures 4.4 and 4.5 respectively.

4.5.4 CBR Usability Evaluation

Our main motivation for two-part text segmentation is that of making documents such as incident reports usable for Case-based Reasoning. Though our approach is able to achieve significant accuracies in splitting two-part documents into the component problem and solution parts, how much of such improvements would translate to improvements in a CBR system needs to be quantified separately. For each problem document, we use a CBR-style retrieval to retrieve the top-3 solutions that are deemed to be usable for the problem from among solutions of other problems in the dataset. The *tot* and *max* metrics measure the total and maximum of the actual usabilities of the three retrieved solutions; Section 3.8 provides more details into the computation of these measures. The *tot* and *max* measures are computed using a cosine similarity based assessment of

Table 4.5: CBR Usability Evaluation (*tot* measure)

Dataset	APS	CCS
visa	0.238	**0.252**⋆
health	0.268	**0.296**⋆
agri	0.394	**0.395**
loan	0.369	**0.404**⋆
tourism	0.161	**0.162**
railways	0.546	**0.612**⋆
telecom	0.202	**0.227**⋆
web	0.236	**0.246**
Average	0.302	**0.324**

the similarities between the proposed solutions and the actual solutions of the posed problem. Cosine similarity looks for identical words occuring across solutions and hence may inaccurately underestimate the usability of the proposed solutions if they use different and semantically similar words, it may be understood as a lower bound of the actual usability. The evaluation measure is averaged across all problems in the dataset so that the performance of a technique on a dataset may be quantified in a single value.

Table 4.5 summarizes the relative performance of the CCS and APS techniques on the *tot* measure, across datasets. It may be noted that *tot*, unlike *WindowDiff*, is not a penalty measure and the values are directly related to be goodness of the technique. CCS is seen to perform better on each dataset whereas the improvement is marginal on some datasets like *agri* and *tourism*; however, significant improvements are recorded for datasets such as *loan* and *railways*. CCS performance was seen to be statistically significant on 5 of 9 datasets, as indicated by the $\star$s in Table 4.5. The trends on the *max* measure was also seen to be similar, with CCS being better than APS on all datasets and recording an average of 7% improvements; statistically significant improvements were achieved on 4 datasets.

4.5.5 CCS Specific Analyses

Having established the improved performance of CCS for segmentation of two-part text documents, we now analyze the CCS technique regarding various factors such as initialization, convergence and goodness of learnt translation models.

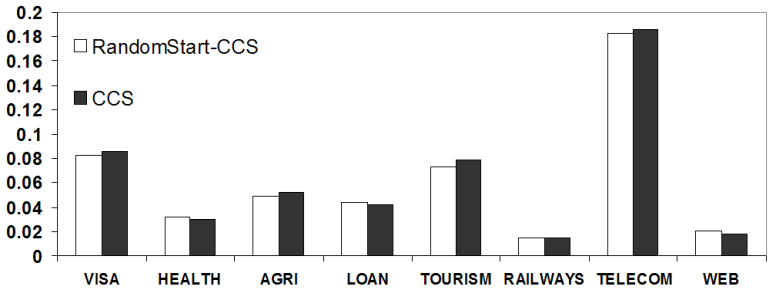

Figure 4.6: CCS Initialization Analysis

CCS Initialization: Since CCS starts off with an APS initialized segmentation vector, part of the performance of CCS may be attributed to the APS initialization. We performed an analysis to estimate the sensitivity of CCS to the initialization quality. Towards this, we compared two versions of CCS, one being the version with the usual APS initialization and the other where the segmentation vector is initialized using random values. The relative performance is plotted in Figure 4.6; somewhat surprisingly, the performance is seen to be nearly identical with minor variations. On some datasets such as *visa* and *tourism*, the randomly initialized CCS is seen to perform marginally better, though the improvement is very minor and could mostly be due to noise effects. This indicates that CCS is extremely noise-robust to initialization. However, a good initialization could become critical for corpora of very long documents.

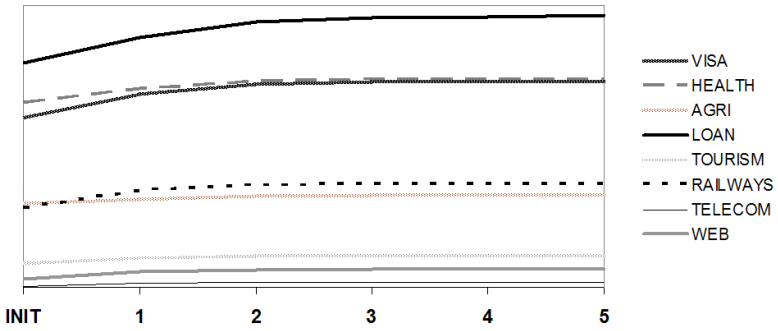

Figure 4.7: CCS Objective Function across Iterations

CCS Convergence: The EM formulation guarantees that the objective function $p(C|\theta)$ would improve with every CCS iteration. The number of iterations it takes for CCS to reach a pretty good value is still important since it would help assess the usability of CCS in cases where quick results is an important consideration. We plot the objective function against #iterations (X-axis) in Figure 4.7. As may be expected for any iterative algorithm, the quantum of objective function improvements diminishes with the number of iterations. For CCS, the objective function is seen to reach a respectable value by the third iteration, with the gains in subsequent iterations being very

Table 4.6: Translation Model Samples from *railways*

Problem Word	*cancel*	*duplicate*	*delivery*
Correlated Solution Words	*tdr*	*deducted*	*letter*
	etktcanc	*misplaced*	*authority*
	slip	*authentic*	*authorization*
	printed	*genuineness*	*cities*

marginal. Thus, for cases where time and computational expense is of much concern, CCS may be terminated after the third iteration since no significant improvements are observed beyond that for the objective function (and thus, presumably, for the $\mathcal{Z}$ vector too).

Example Translation Model Estimates: Having observed from experimental evaluation that usage of the translation model in CCS is able to deliver significant improvements (over CCS$_L$), we now turn our attention to qualitatively assessing the meaningfulness of the correlations learnt by the translation model. Since it is not easy to assess meaningfulness without much domain expertise, we focus on a domain that we are familiar with viz., the *railways* domain. The *railways* dataset contains problems that have to do with ticketing and traveling in the Indian Railways network. Common problems faced by customers of Indian Railways pertain to booking, cancellation and delivery of tickets. We pick three common words from among problems, *cancel*, *duplicate* and *delivery*, and check for the top-10 correlated solution words according to the translation model. Among the 10 solution words, we select the four most understandable words (according to our domain knowledge) and list them in Table 4.6.

For problems related to the word **cancel** that have mostly to do with ticket cancellations, we now try to illustrate why the top correlated words may actually make sense. Ticket cancellation in Indian Railways can be done either online or by walking in to a ticket reservation counter. *etktcanc* represents the online cancellation process whereas cancellation requests need to go through the *tdr* process if the cancellation needs to be done after the scheduled journey commences. For the walk-in cancellation process, one needs a cancellation *slip*, of which the soft copy available on the web can be *printed* out. The other common scenario in the case of offline tickets is that of getting **duplicate** tickets; this involves reporting that the original ticket has been *misplaced* and providing some proof so that the *authenticity* and *genuineness* of the request may be verified.

Issuing of the duplicate ticket also involved *deduction* of some fee from the user. The **delivery** related complaints are often due to the delivery agent refusing to deliver the physical ticket at the user's residence when the user is not present; the reason is most commonly due to the absence of a *letter* of *authorization* being produced by whoever is willing to receive the ticket in the absence of the addressed user. Other inquiries with respect to delivery relate to unavailability of the delivery service in certain *cities*.

4.5.6 Evaluation of Robustness to Noise

CCS derives its improved segmentations due to the availability of a corpus of similar documents, so that the $\mathcal{P}, \mathcal{S}, \mathcal{T}$ models may be learnt over a large collection of documents. The presence of some documents that do not adhere to the assumptions in Section 4.4.1 in the corpus could corrupt the models, and thus induce a bad segmentation on documents that are well-formed (e.g., problem occurs before the solution etc.). On the contrary, the per-document segmentation approaches such as APS produce segmentations that do not depend on the goodness of other documents that need to be segmented. We now analyze the robustness of CCS to cases where some documents do not adhere to the separation and segment ordering assumptions laid out in Section 4.4.1.

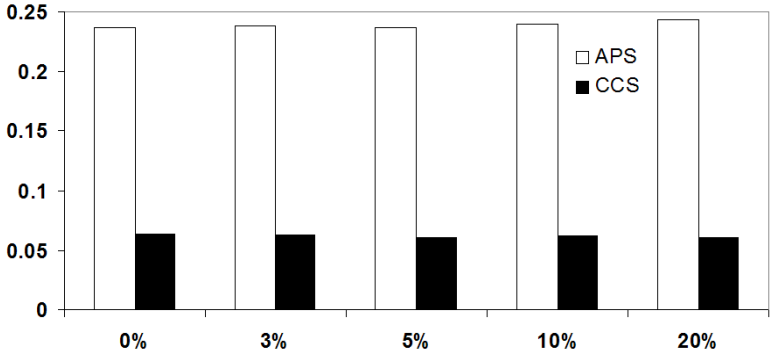

Figure 4.8: *WindowDiff* measure under varying levels of *SentenceSwapping*

Separation: The separation assumption assumes that the problem and solution segments do not appear interleaved in the document, and that there is one single segmentation point that puts the problem and solution sentences at either sides. To evaluate the robustness to this assumption not holding, we inject some noise into documents by swapping sentences on either sides of the true segment boundary, i.e., swapping the last sentence in the problem with the first sentence in the solution. We call this noise as *SentenceSwapping* to convey the kind of noise being introduced. We inject *Sen-*

tenceSwapping noise into different proportions of documents in the corpus, i.e., 3%, 5%, 10% and 20% to create corpora with varying levels of noise. We then run both APS and CCS on them and plot the *WindowDiff* measure under such varying degrees of noise in Figure 4.8. Since the segment boundary is not well-defined for the noise injected documents, we will evaluate the *WindowDiff* only on those documents where the noise has not been introduced. We expect the *WindowDiff* measure to deteriorate with increasing noise for the CCS approach due to such swapping affecting the quality of the models that get generated.

It can be seen from Figure 4.8, CCS is not very much affected by the noise and its performance is seen to be very stable with increasing proportions of noisy documents even upto 20%.

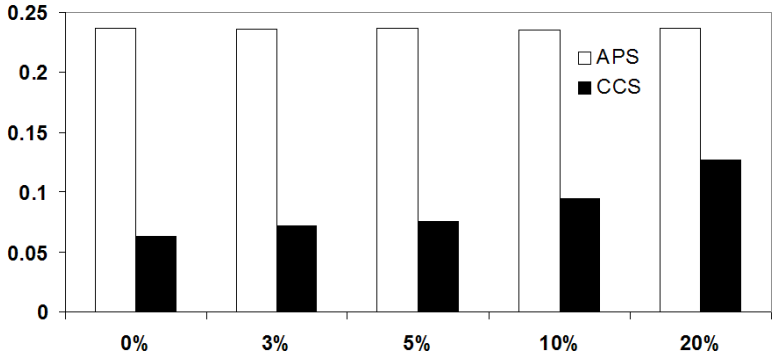

Figure 4.9: *WindowDiff* measure under varying levels of *SegmentSwapping*

Segment Ordering: The segment ordering assumption assumes that the problem segment always precedes the solution segment across all documents in the corpus. Along the lines of the methodology used for testing robustness of the separation assumption, we violate the ordering assumption by swapping the problem and solution segments in different proportions of the documents in the corpus. This kind of noise, *SegmentSwapping*, is more detrimental to CCS since it swaps whole segments instead of just sentences, and has the potential to contaminate the models much more than *SentenceSwapping*. Unlike *SentenceSwapping*, *SegmentSwapped* documents still have a well-defined segment boundary and hence the *WindowDiff* can be calculated over all the documents (and not just the non-noisy ones as done for the earlier case). APS is expected to be least affected by the noise since it works on a per-document basis, and the relative ordering of the segments does not matter to it. We plot the comparative performance of APS and CCS on *WindowDiff* at varying levels of noise in Figure 4.9.

Unlike what was observed for *SentenceSwapping*, CCS is seen to be fairly sensitive to *SegmentSwapping* noise, with the deterioration in performance being significant at noise levels of 20%. *WindowDiff* dips from 0.063 to 0.13 when the fraction of *SegmentSwapping* noise is increased from 0% to 20%. However, CCS is seen to perform much better than APS even at noise levels of 20%, which is encouraging.

4.6 Chapter Summary

We considered the problem of segmentation of two-part text documents into the component problem and solution parts, in this chapter. Accurate identification of the problem and solution segments is a pre-requisite to enable Case-based Reasoning over a large number of document types such as incident reports, diagnosis reports and legal case documents. We argued that two-part text document segmentation poses an unfriendly scenario to many of the state-of-the-art text segmentation algorithms due to the lexical relatedness between the problem and solution parts in such documents. We proposed modeling of such lexical relatedness using translation models, and proposed a technique that uses both language and translation models in addressing the problem. Our technique is designed for cases when a large number of documents from a specific domain are available so that there is enough redundancy in the corpus to learn the language and translation models. We proposed a generative model that samples problem words from a language model learnt over problems, and solution words from either the solution language model or the translation model conditioned on already chosen problem words. Our technique, CCS, was observed to vastly outperform the state-of-the-art methods on well-known segmentation evaluation measures, for the two-part segmentation problem over a large collection of real-world text datasets. Such improved accuracy in segmentation was also seen to translate to moderate improvements in solution usability for Case-based Reasoning systems. We also empirically observed that CCS is relatively tolerant to certain kinds of noise, whereas it degrades slowly and gracefully on other kinds of noise.

We employed IBM Model 1 translation model in CCS; the incremental improvement that can be gained by usage of more sophisticated translation models could be insightful to quantify. Generalizing the two-part text segmentation problem to a general

problem-solution pair identification problem from general text datasets would be interesting future work, and would be a large step towards the goal of enabling Case-based Reasoning over the myriad kinds of text data available from the web and enterprises.

CHAPTER 5

QUERY SUGGESTIONS FOR TEXTUAL PROBLEM-SOLUTION DATASETS

5.1 Introduction

In any retrieval task over text documents, the user typically enters a query on a search box and then clicks submit wherein the processing of the query to discover relevant documents would begin. The most widely used retrieval systems are web search engines, and over the last decade, search engines have become the primary method of finding relevant content on the web. Web search engines exploit properties of web pages, such as existence of links, HTML formatting, anchor texts[1] to rank web pages with respect to a user query; popular ranking algorithms include HITS (Kleinberg, 1999) and PageRank (Brin and Page, 1998). Off late, there has been an increasing trend towards providing query support in forms such as *query expansions* (Fonseca *et al.*, 2005), *query suggestions* (Cucerzan and White, 2007), and ambitious AI-based projects trying to relieve the user from querying efforts altogether (e.g., queryless search[2]). Over the last many years, querying support mechanisms such as query expansion and query suggestions have become commonplace in all popular search engines including Google, Bing, Baidu and Yandex.

For specialized needs, search may have to be run over various kinds of data other than web pages. There are a large set of resources that may not have web-page like characteristics. Common examples include images, patent documents and legal case records. In addition, it has been found that web search like ranking does not work well in scenarios where document creation and linking is not web-like; for example, people do not freely create web pages in an enterprise intranet (Fagin *et al.*, 2003) and the existence of a link to a page does not necessarily indicate endorsement of the destination page. There exist specialized search engines for such specialized search needs;

[1] http://en.wikipedia.org/wiki/Anchor_text accessed February 5^{th}, 2014

[2] http://searchengineland.com/google-offers-queryless-search-personalized-recommendations-10999 accessed February 5^{th}, 2014

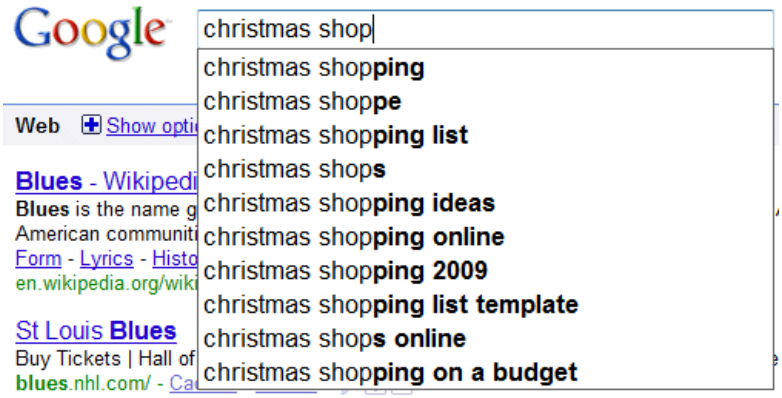

Figure 5.1: Google's Query Suggestion Feature in Action

Wikipedia[3] provides a search feature to search over encyclopedic documents whereas BoardReader[4] provides search over discussion forums. Textual problem-solution repositories, the focus of this thesis, is yet another important resource over which querying is necessary many a time. CBR systems often assume that a (possibly incomplete) specification of a new problem is available so that the retrieval phase may be able to retrieve similar problems from the problem-solution repository. For search-like retrieval over textual problem-solution documents, the problem specification may be sparse enough to be something like a few keywords or a phrase; for example, *sri lankan visa for indians* is a sample specification of a query that represents the need of information pertaining to obtaining a sri lankan visa for indian nationals. Thus, this part of the retrieval problem is more like *information retrieval* than classical methods of retrieval in CBR that are more oriented towards and built for structured cases. A recent work (Xue *et al.*, 2008) proposes a retrieval model for textual problem-solution repositories that incorporates awareness of the problem-solution partition in processing queries; significant gains were reported with the usage of such a model, on various IR evaluation measures. In this chapter, we focus on building a *query suggestion* mechanism similarly tailored towards enabling better retrieval over problem-solution datasets.

As outlined in Section 2.3, textual problem solution repositories are available from a variety of sources. In cases where the problem solution partition is not known, techniques such as those proposed in Chapter 4 or variants may be used. For query systems over textual problem-solution repositories, unless it is the case of an upgrade of an existing widely used search system, usage histories such as query logs are expected to be

[3]http://en.wikipedia.org accessed February 5^{th}, 2014
[4]http://www.boardreader.com accessed February 5^{th}, 2014

very few or even non-existent. Even after long periods of usage of specialized search systems such as those on problem-solution repositories, query logs may not accumulate enough redundancy to be able to depend on as the sole source for deriving query suggestions since the user base is expected to be low, due to the specialized nature of the system and the consequent small user base. Additionally, advanced features such as query suggestions cannot wait until the user base accumulates since those precise features may be necessary to accelerate adoption of such systems. Having laid out the context, we are now in a position to state the problem that we address in this chapter crisply.

Problem Statement: The problem that we address in this chapter is that of generating *query suggestions* for *search systems that operate over textual problem solution repositories* where historical search information such as *query logs are unavailable or scarce*.

We now illustrate certain scenarios and context to argue as to why we think that problem-solution aware query suggestions can improve upon query suggestion mechanisms designed for information retrieval systems that work over regular text datasets. The short motivation section will then be followed by a detailed description of the technique we propose, and its evaluation.

5.1.1 Motivation: Why specialized Query Suggestions for Textual Problem Solution Repositories

We now outline three reasons as to why specialized query suggestion mechanisms are necessary while processing queries over problem-solution repositories. In particular, we outline arguments and scenarios that motivate how we could leverage the problem-solution partition to deliver better query suggestions for our case of querying over textual problem-solution datasets.

Usage of Problem-Solution partition in Retrieval: As outlined in Section 5.1, usage of the problem-solution partition has been shown to improve upon generic retrieval mechanisms (i.e., those designed for general text data). Substantial improvements of upto 25% (Xue *et al.*, 2008) have been recorded. The QA-aware[5] ranking function has

[5]Shorthand for problem-solution partition aware

been described in detail in Section 2.5.5. In the context of such related work from literature, we conjecture that considering the problem-solution partition can deliver better query suggestions too.

Usage of a Query System over QA documents[6]: Consider the usage of a typical IR system that runs over QA documents; the user would query such a system to find solutions to a problem which she would express in a few words or phrases. The need of the user is less of *finding more information about the concept described in the query* and more of *finding solutions to the problem described in the query*. Generic query suggesters for IR such as those described in (Bhatia *et al.*, 2011) intend to discover candidate queries that are *well covered* in the corpus, and such suggestions need not necessarily correspond to queries that represent *well-solved* problems. Consider the sample query *"android emulator"* posed to a corpus where most documents concerning android emulators relate to downloading one such emulator. Thus, a frequency-driven query suggester would intuitively rank a (suggestion) phrase such as *"android emulator download"* highly. However, for this specific case of android emulators, it so happens that android emulators are platform specific; thus, queries that relate to *"android emulators download for windows"* have significantly different solutions than queries that relate to *"android emulators download for linux"*. Thus, we argue that it would be better for the query suggester to suggest phrases such as *"android emulators download for (windows/linux)"* upfront, to avoid repetitive query refinement by going through an intermediate query such as *"android emulator download"*. By being QA-aware, a query suggester could avoid such **underspecification**. We illustrate this example in Figure 5.2.

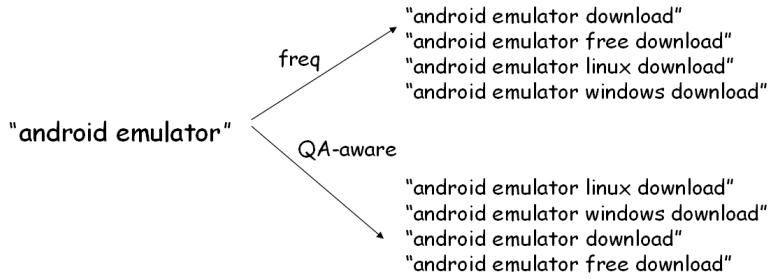

Figure 5.2: Motivating Example of Underspecification Avoidance

An analogous case is the possibility of **overspecification** avoidance in QA-aware

[6]We will use QA documents as a short hand for textual problem solution documents for the sake of brevity

which we will illustrate now. Consider the query *"jellybean"* in a corpus of QA documents where there are numerous problems pertaining to *"jellybean update"*. It could so be the case there is no satisfactory solution to problems involving the latter phrase, leading to a high scatter of solutions for queries relating to it. The QA-aware suggester can de-prioritize *"jellybean update"* due to the high scatter in the solution space whereas the frequency-based suggester is likely to include the phrase in the suggestion due to frequency reasons.

Avoiding "Useless" Suggestions: Consider the query *"phone"* in a corpus of QA documents where the solutions to a large variety of problems involve making a phone call to the contact center. Thus, a frequency-based suggester could end up suggesting *"phone call"* as a potential suggestion even when the phrase is frequent among solutions and not among problems, whereas the user's need is likely to be to find (solutions to) problems that relate to the *phone*. QA-aware suggesters can evidently handle such cases due to being aware of the problem-solution partition.

Despite the above arguments, frequency-driven scoring is definitely useful since it prefers phrases where the user would have more documents to peruse. However, the above reasons suggest that frequency-based prioritization needs to be blended with various other factors to enable arriving at better query suggestions for the user, while querying over textual problem-solution repositories.

5.2 Related Work and Background

Our problem of query suggestions for supporting queries over repositories of textual problem-solution documents and the solution we develop relates to and draws concepts from three broad areas, (1) Query Suggestions for general IR, (2) Retrieval over textual problem-solution repositories, and (3) Textual Case-based Reasoning.

Query Suggestions for general IR: We had presented a summary of work on query suggestions for general IR systems in Section 2.5.6; we recap the main points briefly here. Query Suggestions are different from Interactive Query Refinement (Fonseca *et al.*, 2005) in that the latter deals with providing specializations of a fully-specified information need to enable the user drill down to her precise problem, whereas the former deals with generating meaningful queries from incomplete specifications of information

needs. The difference in nature of the query input has led to development of divergent techniques to tackle these problems. The most popular and well-exploited resource for query suggestions are usage histories such as query logs. Techniques include building a query flow graph (Boldi *et al.*, 2009) from query logs and discovering similar queries based on landing page correlations (Cucerzan and White, 2007). However, our problem is focused on scenarios where usage data is scarce, and thus, the more relevant techniques for our work are those that are able to work in the absence of query logs. As outlined in Section 2.5.6, CompleteSearch (Bast, 2006) and BMM[7] (Bhatia *et al.*, 2011) are two such techniques. The BMM technique relies on two factors for scoring phrases as candidate query suggestions:

1. The correlation to the query estimated using the number of documents they co-occur in.

2. The IDF scores of the words in the candidate query suggestion.

Certain other search systems that restrict the scope the query to some specific types of entities also work without query logs. Examples include product name completions in the search query box in e-commerce marketplaces like Ebay[8], and person name auto-completion in social network search such as those used in Facebook[9]. Since we do not intend to specialize the suggestion to include only names from a specific dictionary such as a person name or product name dictionary, these techniques are outside our radar when it comes to empirical evaluation of the technique we propose.

Retrieval over Textual Problem-solution Repositories: Recently, there has been interest in the Information Retrieval community in developing specialized techniques for scoring problem-solution documents in response to user queries. This led to the scoring function outlined in Section 2.5.5, as was proposed in (Xue *et al.*, 2008). Since then, there has been research in scoring problems from a set of QA documents with respect to similarity with a well-specified user-posed problem; this is evidently different from our problem since we only have a few keywords worth of data to start with (and not the full problem). Techniques for finding similar problems have proposed usage of

[7]Derived from the initials of the authors.
[8]http://www.ebay.com accessed February 5^{th}, 2014
[9]http://www.facebook.com accessed February 5^{th}, 2014

translation model variants such as topic-enhanced (Zhou *et al.*, 2011b) and phrase-level models (Zhou *et al.*, 2011a).

Case Base Alignment Measures: In the technique we develop in this chapter, we exploit some concepts from case-base alignment techniques. Textual CBR relies on the reusability of similar solutions for similar problems. Case-base alignment measures (Raghunandan *et al.*, 2008) quantify how well such assumptions hold, in a specific case base. The case-base alignment measure proposed in (Massie *et al.*, 2007) computes, for each problem, the alignment of the neighborhood of that problem. Consider the case t in Figure 5.3 where the problems of all cases are plotted in the top-half and the corresponding solutions are plotted in the bottom half. It is useful to note that the distance between two entities is meant to illustrate the dissimilarity between them; thus, if two problems are very dissimilar, they would appear very far apart in the top half, and so for the solution space as well. The problem marked as x in the problem space has its corresponding solution marked similarly in the solution space. For the problem t in the problem space, the 3 nearest neighbors happen to be the problems in the cases marked as 2, 1 and 4. Their corresponding solutions are seen to be also close to the solution of t as illustrated in the figure. The local case-base alignment for t, according to the formulation proposed in Massie et. al.(Massie *et al.*, 2007) would then be the following:

$$CB_{\mathcal{D}}(t) = \frac{S_p(t,1) \times S_s(t,1) + S_p(t,2) \times S_s(t,2) + S_p(t,4) \times S_s(t,4)}{S_p(t,1) + S_p(t,2) + S_p(t,4)}$$

where $S_p(.,.)$ and $S_s(.,.)$ denote the similarities between the corresponding problems and solutions respectively, which may be computed using traditional text similarity metrics such as cosine similarity of *tf-idf* metrics. The weighted sum of the similarities of the solutions corresponding the top-k nearest problems to t's problem, are taken in the numerator, and then normalized by the sum of the weights in the denominator. Thus, the local case-base alignment measure of any case d is:

$$CA_{\mathcal{D}}(d) = \frac{\sum_{d' \in kNN(p(d))} S_s(d,d') \times S_p(d,d')}{\sum_{d' \in kNN(p(d))} S_p(d,d')}$$

where $kNN(p(d))$ denotes the top-k similar problems to the problem in d.

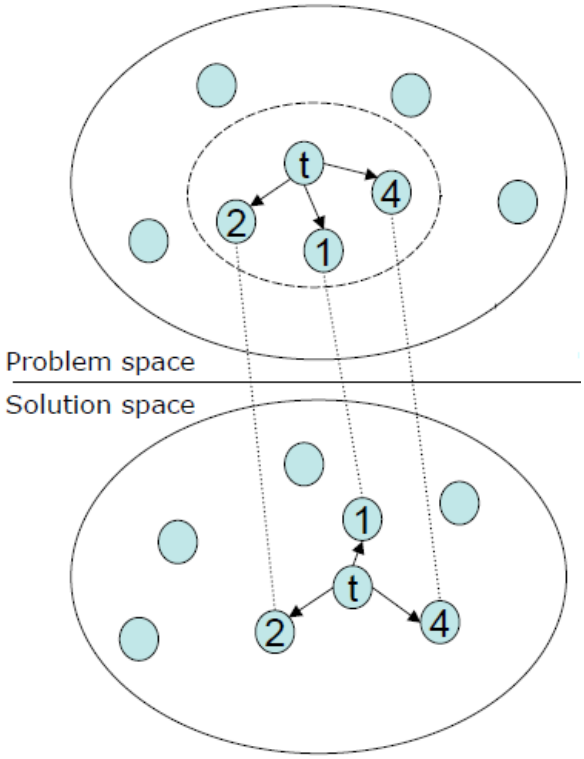

Figure 5.3: Example to illustrate Case-base Alignment

This is averaged across cases in the case base, $\mathcal{D}$, to arrive at a single score signifying alignment of the total case base.

$$CA(\mathcal{D}) = \frac{\sum_{d \in \mathcal{D}} CA_{\mathcal{D}}(d)}{|\mathcal{D}|}$$

Higher values of $CA(\mathcal{D})$ signify that the *similar problems have similar solutions* hypothesis holds better in the corresponding case base. We will use a measure similar to the above local alignment measure, to score cases, in our technique for generating query suggestions.

5.3 Problem Definition

Consider a retrieval system that operates on a corpus of textual problem-solution documents denoted as $\mathcal{D}$. The component documents are represented as $d_i = \{p_i, s_i\}$ where p_i and s_i represent the problem and solution part respectively.

$$\mathcal{D} = [\{p_1, s_1\}, \{p_2, s_2\}, \ldots, \{p_n, s_n\}]$$

For operation of a query suggestion engine, similar to that presented in (Bhatia *et al.*, 2011), we will need to fetch a set of word n-grams from $\mathcal{D}$ as part of a pre-processing operation. A word n-gram is a phrase comprising of n contiguous words; for example $world\ cup$ is a word 2-gram. We will fetch n-grams of length up to 5, in our technique. Let the set of m phrases thus obtained from $\mathcal{D}$ be:

$$\mathcal{P} = \{p'_1, p'_2, \ldots, p'_m\}$$

For an incomplete query from the user, that we denote as Q, we would like to retrieve a set of k phrases from among those in $\mathcal{P}$ such that the estimated likelihood of those phrases being the best completions of Q is maximized. Towards this, we develop a scoring function $\mathcal{S}(p, Q, \mathcal{D})$ that scores each phrase $p \in \mathcal{P}$ with respect to its relevance as a completion for the user input Q for a retrieval system that operates on the corpus $\mathcal{D}$. Once $\mathcal{S}(.,.,.)$ is specified, the top-k phrases would be intuitively derived as follows:

$$top{-}k(Q, \mathcal{D}, \mathcal{P}) = \{p' | p' \in \mathcal{P} \wedge\ \nexists \mathcal{P}* \subset \mathcal{P} s.t((|\mathcal{P}*| \geq k)\ \wedge\ \forall_{p^* \in \mathcal{P}*} \mathcal{S}(p^*, Q, \mathcal{D}) > \mathcal{S}(p, Q, \mathcal{D}))\}$$

The above (slightly awkward) declarative form specifies the top-k phrases as comprising of those such that there are no set of k or more phrases that are better than them under the scoring function $\mathcal{S}$.

5.4 QuerySuggest-QA

We now outline the technique we propose for query suggestions for retrieval systems that operate over problem solution repositories. Given the problem definition outlined in Section 5.3, the specification of our technique is reduced to a full specification of the scoring function $\mathcal{S}(p', Q, \mathcal{D})$ that estimates the likelihood of p being a useful completion for Q. Our scoring function, $\mathcal{S}(p', Q, \mathcal{D})$ is comprised of two parts:

- $P_f(p', \mathcal{D})$: This component is independent of the user input, and ranks each phrase according to its relative frequency in the corpus. This part is mostly modeled according to the query suggestion approach in (Bhatia *et al.*, 2011).

- $P_{\mathcal{D}}(p'|Q)$: This part models the main QA-aware component and calculates the probability of choosing the phrase p as the completion for Q. We will delve into further details of this component in a later section.

These terms are then composed using a product formulation as follows:

$$\mathcal{S}(p', Q, \mathcal{D}) = P_f(p', \mathcal{D}) \times P_{\mathcal{D}}(p'|Q)$$

We now introduce a parameter, λ, that we will use in both the components of the scoring function. λ is used as a relative weighting between the problem and solution parts. Since the user input is expected to be lexically closer to the problem part behavior in $\mathcal{D}$ and so are the query suggestions, we will give more weighting to the problem part while scoring candidate phrases as query suggestions. When a value is specified for λ, the weighting for the problem and solution parts would be λ and $(1 - \lambda)$ respectively; we would like to give a higher weighting for the problem part, so we will use a value of λ that is greater than 0.5. Since we will use λ even in the first part of the scoring function, the first part may also be considered to be slightly QA-aware. We will now describe the two parts of our scoring function in separate sections herein.

5.4.1 Relative Phrase Frequency

In this first component, we will score phrases according to their relative frequency in the corpus $\mathcal{D}$; phrases that occur more frequently would hence have a higher value of $P_f(p', \mathcal{D})$. Since our scoring function needs to rank phrases of different lengths (i.e., 2-grams, 3-grams etc.), we would need to be a bit careful not to give undue advantage to a phrase just due to it having fewer number of terms. A raw frequency estimate is expected to assign higher scores to smaller phrases since they are statistically expected to be more abundant. For example, the phrase $world\ cup$ occurs with at least as much frequency as $cricket\ world\ cup$ since the former is contained in the latter. Thus, we

need to *normalize* frequencies of phrases based on their length. We will use the scheme proposed in (Bhatia *et al.*, 2011) to derive a normalized phrase frequency,$f(p, \mathcal{D})$, as:

$$f(p', \mathcal{D}) = \frac{\#times(p', \mathcal{D})}{log\{avg\{\#times(p'', \mathcal{D})|p'' \in \mathcal{P} \wedge size(p'') = size(p')\}\}}$$

where $size(p)$ denotes the number of words in p. This normalizes the phrase frequency by the log of the average frequencies of all phrases of the same length. Thus, the frequency of a 2-gram is normalized by the log of the average frequencies of all 2-grams in $\mathcal{P}$. We model the $\#times(.,.)$ term as the sum of the weighted frequency using the wieghting parameter introduced in the previous section. Thus,

$$\#times(p', \mathcal{D}) = \lambda \times \#times(p', \{p_1, \ldots, p_n\}) + (1 - \lambda) \times \#times(p', \{s_1, \ldots, s_n\})$$

where $\#times(p', S)$ denotes the number of entries in S that contain p. Having defined the normalized phrase frequency, the $P_f(.,.)$ term is now computed as the $f(.,.)$ score of the phrase normalized by the sum of frequencies across phrases in $\mathcal{P}$ (so that $P_f(., \mathcal{D})$ is a true probability distribution over phrases in $\mathcal{P}$):

$$P_f(p', \mathcal{D}) = \frac{f(p', \mathcal{D})}{\sum_{p'' \in \mathcal{P}} f(p'', \mathcal{D})}$$

5.4.2 QA-Aware Phrase Probability

In modeling the second term that is QA-aware, we will exploit Case-based Reasoning motivated alignment measures to take the correlations between problems and solutions into account. We first start with re-writing the expression using Bayes theorem as follows:

$$P_{\mathcal{D}}(p'|Q) = \frac{P_{\mathcal{D}}(p') \times P_{\mathcal{D}}(Q|p')}{P_{\mathcal{D}}(Q)}$$

Since we are interested in scoring all phrases in $\mathcal{P}$ relative to the user input Q, we

can remove the denominator from the formulation since that depends only on Q and is thus the same for all phrases in $\mathcal{P}$.

$$\approx P_{\mathcal{D}}(p') \times P_{\mathcal{D}}(Q|p')$$

The second term in the right-hand side can now be marginalized over the documents in $\mathcal{D}$ to yield:

$$= P_{\mathcal{D}}(p') \times \sum_{d \in \mathcal{D}} P_{\mathcal{D}}(Q, d|p')$$

We now perform a series of simplifications that are intuitive, as illustrated in the following steps:

$$= P_{\mathcal{D}}(p') \times \sum_{d \in \mathcal{D}} \frac{P_{\mathcal{D}}(Q, d, p')}{P_{\mathcal{D}}(p')}$$

$$= \cancel{P_{\mathcal{D}}(p')} \times \sum_{d \in \mathcal{D}} \frac{P_{\mathcal{D}}(Q, d, p')}{\cancel{P_{\mathcal{D}}(p')}} \qquad \text{(canceling out)}$$

We now apply the chain rule to re-write the equation as:

$$= \sum_{d \in \mathcal{D}} P_{\mathcal{D}}(Q|d, p') \times P_{\mathcal{D}}(d) \times P_{\mathcal{D}}(p'|d)$$

Now, applying the Bayes rule on the first term, we get:

$$= \sum_{d \in \mathcal{D}} \frac{P_{\mathcal{D}}(Q|p') \times P_{\mathcal{D}}(d|Q, p')}{P_{\mathcal{D}}(d|p')} \times P_{\mathcal{D}}(d) \times P_{\mathcal{D}}(p'|d)$$

Most query suggestion systems (Bast, 2006; Bhatia *et al.*, 2011) consider only those phrases that:

- Contain all the full words in Q, i.e., for $Q = "world\ cup\ cri"$, only phrases that

contain both the words *world* and *cup* would be considered.

- Contain a completion of any incomplete word in Q, i.e., for $Q = "world\, cup\, cri"$, only phrases that contain a word starting with cri (e.g., *cricket*, *critique*) would be considered.

Given such filtering of phrases that we will also use in our technique, for a document in which the phrase appears, the query is sure to appear; this is so since the query contains a subset of the words in the phrase (though, not necessarily with the words in the same order). Since p contains Q, $P_{\mathcal{D}}(Q|p')$ evaluates to 1.0 and $P_{\mathcal{D}}(d|p')$ would be the same as $P_{\mathcal{D}}(d|Q,p')$. Thus,

$$= \sum_{d \in \mathcal{D}} \frac{\cancel{P_{\mathcal{D}}(Q|p')} \overset{1.0}{\times} \cancel{P_{\mathcal{D}}(d|Q,p')}}{\cancel{P_{\mathcal{D}}(d|p')}} \times P_{\mathcal{D}}(d) \times P_{\mathcal{D}}(p'|d)$$

Having simplified the formula to just two terms, we can think of the first term as the document importance and the second term as the phrase probability.

$$= \sum_{d \in \mathcal{D}} \underbrace{P_{\mathcal{D}}(d)}_{\text{document importance}} \times \underbrace{P_{\mathcal{D}}(p'|d)}_{\text{phrase probability}}$$

We will now outline how we compute the *document importance* and *phrase probability* in the following paragraphs.

Document Importance: Towards quantifying document importance, we will use local case-base alignment measures such as the ones discussed in Section 5.2. We would like to give a higher weighting to documents in whose vicinity the case-base is aligned well. This is motivated by the fact that areas where a case-base is aligned may be thought of as comprising of *well solved* problems since there are many similar solutions to similar problems. In particular, we would like to prefer a document $d = (q, a)$[10] if:

- *$\mathcal{D}$ contains many problems similar to q*: Unless there are a lot of problems similar to q, we would not be able to estimate how well the problem q is solved. A set of

[10]We use (q,a) instead of (p,s) to avoid confusion, given that we use p' to represent the phrase.

similar problems gives us enough redundancy to assess how well the problem q is solved.

- *Problems in $\mathcal{D}$ similar to q have solutions similar to a:* This is similar to the CBR hypothesis that similar problems have similar solutions; if the hypothesis holds well, we have enough confidence that the problem in q is well-solved in $\mathcal{D}$. Thus, phrases in q may be suggested with high confidence.

To recap, the local case-base alignment measure from Section 5.2 is as follows:

$$CA_{\mathcal{D}}(d) = \frac{\sum_{d' \in kNN(p(d))} S_s(d, d') \times S_p(d, d')}{\sum_{d' \in kNN(p(d))} S_p(d, d')}$$

where $kNN(p(d))$ denotes the top-k similar problems to the problem in d. Consider two case documents where one has a lot of very similar problems, and another where the similar problems are far away (lexically). Both these may evaluate to the same value of the $CA_{\mathcal{D}}(d)$ if the solution similarities of the cases in the respective $kNN(p(d))$s to the corresponding solutions are similar. However, even under such a case, we would like to assign the latter case document a lower weight since its similar problems are further away than the similar problems of the former. Towards this, we just modify the calculation by simply avoiding the denominator in the expression for $CA_{\mathcal{D}}(d)$. Our modified local case-alignment measure, $CA'_{\mathcal{D}}(d)$ now stands as:

$$CA'_{\mathcal{D}}(d) = \sum_{d' \in kNN(p(d))} S_s(d, d') \times S_p(d, d')$$

Our estimate of the document importance, is now the fractional case-base alignment measure, expressed as a fraction of the total local case-base alignment across all documents in $\mathcal{D}$.

$$P_{\mathcal{D}}(d) = \frac{CA'_{\mathcal{D}}(d)}{\sum_{d' \in \mathcal{D}} CA'_{\mathcal{D}}(d')}$$

Phrase Probability: We use the probability of generating the phrase using a unigram language model derived from the document, as the phrase probability, $P_{\mathcal{D}}(p'|d)$. This computation is done as outlined in Section 4.2.4. Here too, we use the weighting

parameter λ leading to a final formula as below:

$$P_{\mathcal{D}}(p'|d) = \lambda \prod_{w \in p'} \frac{count(w,q)}{\sum_{w' \in q} count(w',q)} + (1-\lambda) \prod_{w \in p'} \frac{count(w,a)}{\sum_{w' \in a} count(w',a)}$$

where $count(w,s)$ denotes the frequency of w in the text fragment s.

5.4.3 Combined Formula

Having restricted the search to only those phrases in $\mathcal{P}$ containing Q, the final scoring function may now be written as:

$$\mathcal{S}(p',Q,\mathcal{D}) = \begin{cases} 0, & \text{if p' does not contain Q} \\ P_f(p',\mathcal{D}) \times \sum_{d \in \mathcal{D}} (P_{\mathcal{D}}(d) \times P_{\mathcal{D}}(p'|d)), & \text{otherwise} \end{cases}$$

Informally, we score all phrases that have all complete words in Q (not necessarily in the same order) and valid completion of any incomplete word in Q, according to the above scoring function that blends frequency based scoring and problem-solution correlations. Thereon, we take the phrases with the top-k scores, for presentation to the user as query suggestions.

5.4.4 Time Complexity of QuerySuggest-QA

For every phrase p in $\mathcal{P}$, the value of the following expression may be pre-computed:

$$P_f(p',\mathcal{D}) \times \sum_{d \in \mathcal{D}} (P_{\mathcal{D}}(d) \times P_{\mathcal{D}}(p'|d))$$

This is precisely the expression in the second case in the combined formula in Section 5.4.3. When the user enters a query, the only task now is to find the subset of phrases that contain the query, and rank them according to the pre-computed scores. Even without indexes, this could be done in $\mathcal{O}(|\mathcal{P}| + k\ log\ b)$ where b denotes the

number of phrases in $\mathcal{P}$ that contain Q. A first pass through the $|\mathcal{P}|$ phrases is done to identify the subset of b phrases, after which a partial sort to identify the top-k phrases is performed. This may be speeded up if reverse indexes that are commonly used in information retrieval engines are employed. The check for inclusion of Q in the phrase is deemed to take constant time; this is reasonable since phrases contain up to 5 words, and queries are even smaller, thus leading to only $10s$ of checks for each phrase. When the collection of documents and phrases are dynamic, the pre-computation is no longer feasible, and the computational complexity would go up.

5.5 Experimental Evaluation

We now empirically evaluate QuerySuggest-QA against the BMM technique (Bhatia *et al.*, 2011), the state-of-the-art query suggestion technique that works without query logs. Since the focus of our evaluation is the efficiency of QuerySuggest-QA for retrieval over textual problem-solution repositories, we will use textual problem-solution datasets in our evaluation. We describe the datasets and evaluation measures, followed by a detailed description of our extensive empirical analysis. We set the weighting parameter λ to 0.9 in the QuerySuggest-QA technique, unless mentioned otherwise.

5.5.1 Datasets

Since Information Retrieval work is typically evaluated over large datasets, we choose two of the largest datasets that we have from those described in Section 2.7; *career* and *insurance* that contain 938 and 990 documents respectively. We additionally pick the smaller *sports* collection (357 documents) too, to quantify the improvements achieved by our technique on smaller datasets. These datasets are from an IR task[11] that has corresponding query sets as well. We chose phrases from among noise-corrected queries in the query collection from the IR task. Since we wanted to work with incomplete queries, we chose either the first 1-2 words, or the first 1-2 words along with a prefix of the next word to use as incomplete queries Q. Of the 45 queries we collected, the summary of query distribution appears in Table 5.1.

[11] http://irsi.res.in/fire/faq-retrieval/Resources.html accessed February 5^{th}, 2014

Table 5.1: Dataset and Query Set Summary

Dataset	#Documents	#Queries
career	938	16
insurance	990	17
sports	357	12

The incomplete queries along with suggestions discovered by both the techniques, QuerySuggest-QA and BMM, were given to 1-2 human labelers who were asked to label the query suggestion as being either relevant or irrelevant. There could be potential conflicts between labelers where one labels a suggestion as relevant, whereas the other estimates the suggestion to be irrelevant. Since lack of domain expertise could lead to falsely underestimating relevance, we took the OR of such labelings, thus retaining the *relevant* label for a query suggestion even if only one of them judged it as relevant. We use such labelings to compare the performance of the various techniques on classical IR evaluation measures that we will outline in the following section.

5.5.2 Evaluation Measures

We use traditional rank-based effectiveness evaluation measures (Robertson and Zaragoza, 2007) from the Information Retrieval community, in quantifying our experimental analyses. In particular, we use four IR evaluation measures: MRR, MAP, NDCG and Precision. We briefly explain each of them here:

Mean Reciprocal Rank (MRR): For each query, the reciprocal rank is calculated as the reciprocal of the rank of the first relevant result (e.g., if the first relevant result appears at Rank 3, the reciprocal rank evaluates to 0.33 for that query). This measure is averaged across queries to get the MRR measure across queries.

Mean Average Precision (MAP): The Average Precision of a ranked list of results is the average of the precisions at those positions where a relevant result (i.e., query suggestion) appears. This measure averaged across queries leads to the MAP measure.

Normalized Discounted Cumulative Gain (NDCG): The Discounted Cumulative Gain measures the sum of the relevance of results in the list, discounted by the logarithm of their position in the list; such a formulation helps rank relevant results higher up in

the result list. The Normalized DCG normalizes this measure by the maximum possible value of DCG till the position in the list that we consider.

Precision (Prec): This simple measure quantifies the fraction of relevant results in the list, without taking into account the order in which they appear.

In addition to the above, we also report *Success Rate* (Bhatia *et al.*, 2011), the number of queries for which at least one relevant query suggestion was retrieved in the top-k suggestions. Much like in Section 4.5.2, we use statistical significance assessments using randomization tests against p-values of 0.05 and 0.01 to determine whether the improved performance of the better performing technique is statistically significant.

The traditional IR quality measures outlined above quantify the intristic quality of the query suggestions as estimated by user perception. Being a Textual CBR system, in addition to presenting relevant suggestions, we would also like to check whether the suggested query retrieves documents containing problems that are *well-solved* in the corpus. This is important since we do not want the user to land on documents describing problems that contain only one or very few solutions (from within $\mathcal{D}$); this is done with the intent of minimizing user dissatisfaction. Though we use the term *well-solved problem* a bit loosely, we intend to quantify the number of *usable* solutions from across neighbors of the document, from within $\mathcal{D}$. Fortunately, we have already outlined measures to quantify such notions in Section 3.8, and we can readily use measures such as tot and max off the shelf. For a phrase p that is offered as a query suggestion, we estimate the tot measure as:

$$tot_{\mathcal{D}}(p) = \sum_{d \in IR\_Result(p,\mathcal{D})} tot(d, R(d))$$

where $IR\_Result(p, \mathcal{D})$ is used to refer to the set of documents retrieved by an IR system from $\mathcal{D}$ in response to the query p. $R(d)$ denotes the set of solutions from the CBR system using the corpus $\mathcal{D}$ when posed with the problem part of document d. Further details of how tot is computed appears in Section 3.8; informally, it quantifies (using statistical text similarity measures) the total usability of the solutions retrieved by a CBR system to the problem that triggered the retrieval. tot is an extrinsic evaluation measure that estimates how well the suggested query is able to drive the user towards well-solved problems; since tot is estimated without making use of user input,

Dataset	Precision@10		MAP@10		MRR		NDCG@10	
	BMM	QS-QA	BMM	QS-QA	BMM	QS-QA	BMM	QS-QA
career	0.538	**0.775**$^{\bullet}$	0.688	**0.895**$^{\bullet}$	0.820	**1.0**$^{\bullet}$	0.668	**0.900**$^{\bullet}$
insurance	0.559	**0.694**$^{\circ}$	0.803	**0.839**	0.961	**1.0**$^{\bullet}$	0.818	**0.841**
sports	0.508	**0.617**$^{\circ}$	0.719	**0.805**	0.892	**0.958**$^{\bullet}$	0.728	**0.814**
All	0.538	**0.702**$^{\bullet}$	0.740	**0.850**$^{\bullet}$	0.892	**0.989**$^{\bullet}$	0.741	**0.855**$^{\bullet}$

Table 5.2: Results of Evaluation ($\circ$ & $\bullet$ denote statistical significance at $p < 0.05$ and $p < 0.01$ respectively)

Dataset	Success Rate@1	
	BMM	QS-QA
career	0.75	**1.0**$^{\bullet}$
insurance	0.941	**1.0**$^{\bullet}$
sports	0.833	**0.917**$^{\bullet}$
All	0.844	**0.978**$^{\bullet}$

Table 5.3: Evaluation on Success Rate($\bullet$ denotes statistical significance at $p < 0.01$)

it provides an orthogonal view of the quality of the suggested queries.

The five IR measures and the CBR evaluation measure, tot, are all computed on a per query basis. We aggregate each measure by simply averaging it over all the queries considered leading to a single value of the evaluation measure for each dataset.

5.5.3 Comparison over IR Evaluation Measures

We present the results of the empirical evaluation over the various datasets in Table 5.2. The performance on each evaluation measure for the baseline technique BMM and our technique QuerySuggest-QA (abbreviated as QS-QA) are listed therein. For most of the measures, we compute the performance using k=10 (whereas we use k=1 for success rate in Figure 5.3); this is indicated by the @10 (@1) suffix in the column names in the results table. It can be seen that QS-QA performs consistently better than BMM and sometimes by large margins. QS-QA is particularly efficient over the large datasets *career* and *insurance* where the performance reaches the absolute maximum value of 1.0 on MRR and SR. The impressive gains on the precision values (upto 25%) show that the improvements achieved by QS-QA continue down the list and are not limited to the first 2-3 results; the MAP, MRR and NDCG metrics weigh the top results highly. Coming to the Success Rate comparison in Figure 5.3, it was found that the QS-QA technique retrieves at least one relevant result among the top-2 results; BMM's first relevant re-

sult appears at rank #8 for certain queries. The success rates against varying k across all datasets are plotted in Figure 5.4. The improved performance was also seen to be statistically significant in many cases as illustrated by the bordered dot (corresponding to p-value 0.05) or the solid black dot (p-value 0.01) in Tables 5.2 and 5.3. In summary, QS-QA outperforms BMM by vast and statistically significant margins across datasets.

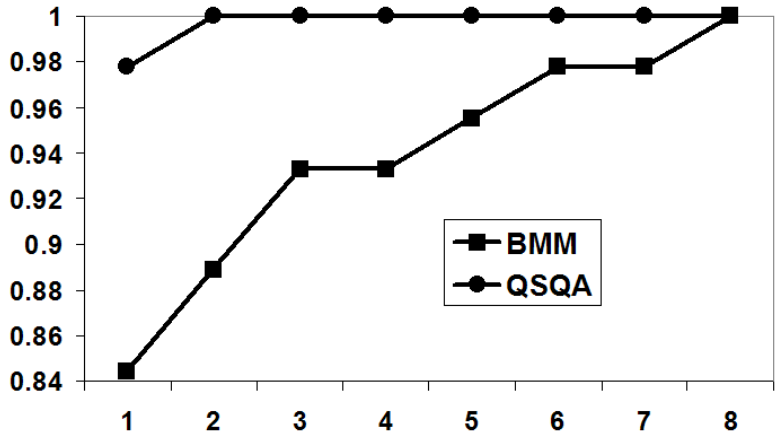

Figure 5.4: Experimental Results: Success Rate with varying k

5.5.4 Comparison on the *tot* Measure

The tot measure is fundamentally different from the IR evaluation measures since it quantifies the extent to which good solutions may be achieved for problems in documents retrieved using the suggested query. Notably, it does not use the manual labelings, and thus provides an orthogonal insight into the effectiveness of the techniques. It was seen that QS-QA outperforms BMM on this measure too, with gains varying between a low of 12% and a high of 19%. The *tot* measure plotted against varying k across all datasets is shown in Figure 5.5.

5.5.5 Sample Results: Qualitative Evaluation

Having illustrated the effectiveness of QS-QA by the comparative evaluation on the various evaluation measures, we now present some sample query suggestions by both the techniques to give an intuitive feel of the nature of the suggestions. Though this is by no means comprehensive, we hope that it serves to illustrate the nature of the improvement achieved by QS-QA. We present a few sample results in Table 5.4 where the relevant phrases (as assessed by human labelers) are marked with a $*$. For the first

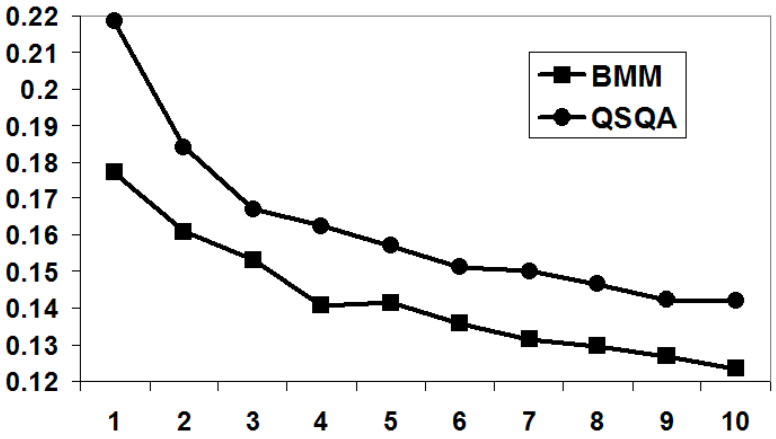

Figure 5.5: Experimental Results: tot with varying k

Domain: *Insurance*, Query: *'card'*		Domain: *Career*, Query: *'engineering de'*	
QuerySuggest-QA	BMM	QuerySuggest-QA	BMM
health card*	health card*	degree in engineering*	engineering in eee dept
credit card*	credit card*	engineering or degree	details marine engineering*
id card*	health card mentions	degree in an engineering	engineering would depend
medicare card*	card mentions the contact	degree in aeronautical engineering*	depend on performance in engineering
card which	health card function	details marine engineering*	degrees in computer engineering*

Table 5.4: Sample Results Comparison ($*$ is marked against suggestions deemed to be relevant by human labelers)

query that comprises of the single word *card*, QS-QA is able to retrieve phrases corresponding to many types of cards regarding which there were problems in the dataset. Thus, *health card*, *credit card*, *id card* and *medicare card* find their place in the top-4 results of QS-QA. On the other hand, only those types of cards that are represented with high frequency in the corpus are retrieved by BMM; the third, fourth and fifth results all pertain to the most frequent referent for *card*, the *health card*. The second query *"engineering de"* is more difficult since the most common referent for the query is the topic *engineering degree*, and this limits the scope for search. However, QS-QA ranks topical phrases higher even in this case, leading to a better performance as illustrated in the results table.

5.5.6 Sensitivity to Weighting Parameter

We have set λ to 0.9 consistently in our evaluation. However, this being a weighting parameter, the sensitivity to the parameter may be of interest in assessing the practicality of using QS-QA in diverse scenarios. Based on an empirical evaluation, our technique was not seen to be very sensitive to λ; the performance remained very stable between 0.5 and 0.9 while gains started to drop gracefully at either ends of the $[0-1]$ spectrum.

5.6 Chapter Summary

In this chapter, we considered the issue of query suggestions for IR-style queries over textual problem-solution repositories. We explored the possibility of utilizing the problem-solution partition inherent in textual case bases to derive more effective query suggestions than generic query suggesters. Our technique is designed to work in the absence of usage information such as query logs; this is a common scenario encountered in the case of usage of specialized search systems. We developed a technique that blends concepts from Case-based Reasoning and language modeling to score candidate suggestions in response to incomplete user queries. In particular, we adapted a local case-base alignment measure, and used that to assign weights to documents in the corpus; phrases from highly weighted documents would then be preferred to those from lower weighted documents, in presenting to the user as candidate query suggestions. We compared our technique against the state of the art query suggestion technique for general IR queries, BMM. Based on an extensive empirical evaluation over real-world datasets, we established the effectiveness of our technique (QS-QA) over BMM, for the specific problem of providing retrieval support over textual problem-solution repositories. QS-QA achieved large and statistically significant improvements over classical IR evaluation measures such as MAP, MRR, NDCG and Precision. Further, the utility of QS-QA in retrieving documents containing well-solved problems was quantified using the tot measure. On this Textual CBR evaluation measure, QS-QA was seen to outperform BMM by margins ranging from 12% to 19%. Thus, in this chapter, we have designed a method for exploiting the problem solution partition in generating query suggestions for IR queries over textual problem solution datasets and empirically established the superiority of our technique over state-of-the-art methods for the problem we address.

Future work in this regard may be directed towards ensuring diversity in query suggestions so that the user may be presented phrases covering a wider range of problems, thus potentially leading to higher success rates in addressing user's information needs. Instead of scoring phrases based on just statistical considerations, it may be useful to subset the search to a specific kind of phrase, for example *noun phrases*, or those that agree to common formats such as *subject-verb-object* and so on.

CHAPTER 6

CBR MOTIVATED PROCESSING OF GENERAL TEXT DATA

6.1 Introduction

The focus of the work presented in this thesis has been processing of problem-solution repositories or datasets having problem-solution components. However, a vast majority of data available on the web may not have such components. To leverage the large amounts of unstructured text documents on the web and the research that has gone into developing techniques to process such content electronically, techniques to enable usage of unstructured text documents (that do not necessarily have a problem-solution partition) in each phase of the CBR process need to be developed. Possibilities of enhancing Textual CBR using text data include the following:

1. **Drill down to Text Data Subsets:** CBR systems are very often domain-specific, and are built using problem-solution data available for the particular domain. With the advent of the web and large scale digitization of enterprise data, plenty of domain-specific text data are available in formats other than in the form of structured problem-solution documents. To enrich a domain-specific CBR system, we would need to be able to quickly drill down to the subset of domain-specific text data of interest as first step. For example, if we are interested in building a medical diagnosis support system, we may have to first identify the subset of medical documents from the large stockpile of enterprise or web data that is available. This relates to the general text mining problem of clustering and/or classification, depending on the availability of labeled data.

2. **Identifying Text containing Problem-solution Information:** Among the domain-specific text data that can be selected using text mining techniques as outlined

above, there could be many text documents that contain problem-solution information. To exploit them within the classical CBR framework, we would first need to identify text segments that contain problems and solutions. This could be accomplished by style-based classification of text segments, where *signatures* of problem and solution texts are used to identify segments that match such signatures. Once such problems and solutions are identified from text data, these would now need to be matched to create problem-solution pairs so that they can be readily fed into the case base. Solution segments may normally be attached with the lexically closest problem segment if they are from the same document; more sophisticated lexical correlation-based matching techniques may be necessary to match problems to their potential solutions from other documents.

3. **Retrieval of Related Text Documents:** Some of the documents from task (1) above may not contain any problem solution information. However, they may be highly similar to the problem part or solution part of some cases in the case base. For scenarios where the newly posed user problem has very few similar problems in the case-base, or in cases where the solutions retrieved in response to a user problem are lexically sparse, it would be beneficial to include very similar domain-specific text documents in the results. Informally, for user problems in the space where the case-base is weakly aligned, the CBR system may be made to evolve gracefully into a IR-like system by providing a mix of CBR and IR results.

4. **Using Social Media Data in Textual CBR:** Social media provides immense possibilities to CBR and recommender systems. *Trusted Recommendations* is one way of harnessing social media data, where recommendations are prioritized if they relate to entities that have been used/visited by "friends" of the querying user. In Community-driven Question Answering type social networks, a user may be provided with solutions authored by users who are close to them in the social connections graph, since she may find such solutions more trustworthy. In addition to connection-based similarity, content based similarity provides an orthogonal means of ranking content; solutions authored by those who generally post similar content (as the user who issues the query) may be prioritized due

to intuitive reasons. Social media content is different from usual text documents in that they could additionally contain video links, pictures and annotations of other users; the text content itself may be very short and studded with uncanny abbreviations such as those used in chat and SMS.

As may be apparent from the ongoing discussion, the possibilities of usage of raw unstructured text data to enhance Textual CBR are many. Two broad themes are the *extraction* of cases from raw text and usage of related text documents as *auxiliary content*. In this chapter, we focus on two specific problems from among the various challenges outlined above. We briefly describe the problems that we address, followed by a detailed description in separate sections.

- **Interpretable Clustering:** The unsupervised text mining technique of clustering helps drill down to domain-specific subsets of text documents from among a vast collection of text documents. Clustering groups text document corpora into groups of coherent documents, such that the documents within a cluster are lexically similar to each other. Clustering is often used in scenarios such as presenting a group of documents where presenting similar documents in a group makes it simpler to assimilate them. Our interest in clustering is motivated by the need to find a small subset of domain specific text documents that could be later processed to enrich a CBR system; thus, we are interested in finding groups of coherent documents, where each group is associated with a description that would enable us to assess whether it is useful for the domain of interest. Most clusters would be rejected when the clustering is performed on a diverse collection since the set of domain-related documents are expected to be small. Interpretable clustering associates human-readable descriptions (such as collections of words) with clusters, and are hence best suited for our problem of assessing relevance to the domain. We develop a technique for interpretable flat clustering that improves upon state-of-the-art interpretable clustering methods.

- **Retrieval in Microblog Data:** Social media data are much more complex than text documents since they encompass different kinds of content such as video

links, images, and are often authored using strange abbreviations. Traditional text similarity metrics look for occurrence of the same words across documents to assess similarity; chat-lingo pose a problem here since the same word may be abbreviated differently (e.g., *uni* or *univty* for *university*). Thus, we argue that better similarity measures tailored to suit social media data would help bringing social media data under the purview of CBR, since similarity-based retrieval is among the major-building blocks of CBR systems. We specifically look to developing similarity measures for microblog data generated from social media sites like Twitter that impose a length restriction on posts.

We discuss these problems and propose novel techniques for addressing them, along with an extensive empirical evaluation to assess their effectiveness. Since these problems are not related to the core CBR processes and have been traditionally addressed in domains such as text processing and data mining, we evaluate them against state-of-the-art methods from respective fields.

6.2 Interpretable Clustering of Document Datasets by deriving Word-based Rules

We now focus on techniques to enhance the understandability of clustering output, in order to easily select or reject whole clusters based on a visual inspection. This is motivated by the need to quickly drill-down to clusters of interest. It could be the case that a specific cluster contains documents from the domain that we are interested in, but, also contains documents from other domains. We would like to attempt to provide some mechanisms to enable the user to drill-down *within* clusters by editing the cluster representation. Thus, we focus on the problem of clustering documents to generate cluster descriptions that are both:

- **Interpretable:** By sifting through a cluster description, a user should be able to assimilate the nature of documents in the cluster. Mechanisms such as Tag Clouds (Halvey and Keane, 2007) provide means to visualize document subsets on the web.

- **Reconfigurable:** A user who is interested in selecting clusters from a particular domain may find that some clusters, despite containing documents from the domain of interest, are contaminated by documents from one or more other domains. In such a case, the user may want to meaningfully edit the cluster, by tweaking the cluster description, to make it a more domain-pure cluster. Such editing is possible in the case of descriptions of classes provided by decision trees where such editing may be done by changing the thresholds in a particular predicate, or by removing or adding predicates altogether.

The state-of-the-art method for interpretable clustering (Basak and Krishnapuram, 2005) provides a hierarchical clustering of document datasets. While the hierarchy conveys additional information that is useful, flat (partitional) clustering has been the paradigm subjected to much more research (Jain *et al.*, 1999); flat clustering is more popular and is the default setting of many clustering toolkits[1].

We focus on the problem of interpretable and reconfigurable flat clustering of document datasets in this section. We develop a technique for *Rule Generating Clustering*, RGC-D and compare it empirically to various other clustering methods on real-world datasets. Earlier versions of the proposed technique were reported in (Balachandran, 2009) and (Balachandran *et al.*, 2009).

6.2.1 Context and Related Work

Clustering, the problem of grouping objects with the intent of generating coherent groups in accordance with a chosen similarity measure, has been a very well-studied problem for the last many decades since the introduction of the K-means algorithm (MacQueen, 1967). While the primary output of clustering is the assignment of objects to groups, different ways of representing a clustering have been proposed. The implicit cluster description from K-means, the per-cluster centroid vector, has been criticized to be not very meaningful to describe clusters (Boley, 1998). Other popular ways of visualizing the content in sets of documents (since clusters are document sets) are (1) Sets of Representative words (as used in Cluto toolkit), and (2) Tag/Word Clouds (Halvey

[1] http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview accessed February 5^{th}, 2014

and Keane, 2007). However, such word-based representations (*WBR*) of clusters are not self-contained and are formed by using a selection of frequent words in the cluster; a document may contain words from across two cluster descriptions, and may belong to a third. At the other extreme, one document may contain none of the words in the description of the cluster that it is member of. Thus, the WBR representation is not a self-contained description of a cluster, and consequently is not editable. That it is not editable follows as a consequence of not being self-contained since operations such as splitting the words in the WBR into two sets do not intuitively define the split in the document space. Outside clustering literature, topic modeling based methods such as LDA (Blei *et al.*, 2003) represent documents as mixtures of topics (topics are more abstract entities than words) to aid understanding, and have been found to be very useful in identifying the predominant topics in a corpus. Though topics provide higher level concepts to aid understanding document corpora, probability distribution based representations of document clusters may not be very friendly for human assimilation (in addition to being non-editable models, like WBRs). On the other hand, there are techniques from the fuzzy systems community that lead to cluster descriptions that are eminently self-contained (and editable). Popular approaches for rule-induction in fuzzy systems include techniques such as RIPPER (Cohen, 1995) and SLIPPER (Cohen and Singer, 1999). They are, however, supervised learning techniques similar to classical decision trees, and are hence not applicable to our problem.

Our problem deals with learning *interpretable* and *self-contained* (correlated with editability) clusters from document corpora in an *unsupervised* fashion. We present a summary of related work in Table 6.1 evaluating the various techniques on these three dimensions. The techniques that suit our problem definition have been listed in bold font, and they are FindClans, CLUSTER/2 and UDT.

FindClans uses a Sum-of-Rectangles description model for clusters, whereby a cluster is described by the a set of rectangles that fully contain all points in the cluster. For example, $(1 \leq x \leq 5) \wedge (3 \leq y \leq 4)$ represents a rectangle in the XY-space and comprises of data points such as $(3, 3)$ and $(4, 3)$. However, text data is inherently extremely multi-dimensional; vocabulary sizes can reach up to thousands, and since words are analogous to attributes, documents are represented in space that has thousands of dimensions leading to a very sparse space. *FindClans*, due to being driven by the idea of representing a set of documents by a set of boxes where the number of boxes need to

Table 6.1: Overview of Related Work

Technique	Unsupervised?	Interpretability	Self-contained?
Decision Tree(Breslow and Aha, 1997)	No	High	Yes
RIPPER(Cohen, 1995)	No	High	Yes
SLIPPER(Cohen and Singer, 1999)	No	High	Yes
LRI(Weiss and Indurkhya, 2000)	No	High	Yes
FindClans(Gao and Ester, 2006)	Yes	High	Yes
CLUSTER/2(Michalski and Stepp, 1983)	Yes	High	Yes
UDT(Basak and Krishnapuram, 2005)	Yes	High	Yes
COBWEB(Fisher, 1987)	Yes	High	No
K-Means(MacQueen, 1967)	Yes	Low	Yes
Tag Clouds(Boley, 1998) & WBRs	Yes	High	No
HAC(Jain *et al.*, 1999)	Yes	Low	Yes

be far fewer than the number of documents, would not scale well into very sparse and high-dimensional spaces. *CLUSTER/2* uses an approach like *FindClans*, but, instead of finding rectangles, restricts itself to discovering intervals such as $1 \leq x \leq 5$. Due to similar reasons as for *FindClans*, *CLUSTER/2* would also be hard to adapt to high-dimensional and sparse spaces. This leaves us with *UDT*, which, as shown in (Basak and Krishnapuram, 2005), can be used to cluster text datasets. UDT is a decision-tree based technique that starts with the entire document corpus as the dataset associated with the root node, and recursively splits nodes to form a hierarchical tree representation. The splitting condition is chosen as the one that leads to the highest information gain, and the split process terminates at nodes that have fewer than a specified number of documents. We will use UDT in our experimental evaluation as a baseline method.

6.2.2 Rule-Generating Clustering with Disjunctive Rules (RGC-D)

We present our interpretable clustering technique, RGC-D, in Algorithm 3. To make the approach easy to understand, we will first outline the three different phases that RGC-D employs, before describing the pseudo code in detail.

Figure 6.1 depicts the process pictorially. RGC-D takes one main parameter, k, that denotes the number of clusters in the output; such a parameter is used in many partitional clustering algorithms including the well-known k-means. The first phrase generates clusters that have associated rules, and could generate more than k clusters. The second phase progressively merges the most similar pair of clusters, until there

Alg. 3 Rule-Generating Clustering with Disjunctive Rules (RGC-D)

Input. $\mathcal{D}$, a set of documents, and k, the desired number of clusters in the output
Output. C, a clustering of documents from $\mathcal{D}$ and R a set of rules with one rule for each cluster in $\mathcal{C}$

Phase 1: Cluster Generation

1. $C \leftarrow \phi$, $R \leftarrow \phi$,$W_\alpha =$ *top-t words acc. to CR*
2. $while\ W_\alpha\ has\ words\ yet\ to\ be\ considered$
3. $\quad w = \arg\max_{w \in W_\alpha} |D_w - C|$
4. $\quad if\ D_w\ is\ disjoint\ with\ existing\ clusters$
5. $\quad\quad C = [C, \{D_w\}], R = [R, \{w\}]$
6. $\quad else\ if\ D_w\ has\ a\ similarity\ of\ at\ least\ \gamma\ with\ all\ overlapping\ clusters$
7. $\quad\quad Merge\ overlapping\ clusters\ with\ D_w\ forming\ a\ new\ cluster$ $whose\ rule\ is\ a\ disjunction\ of\ those\ of\ component\ clusters$

Phase 2: Cluster Merging

8. $while\ |C| > k$
9. $\quad pick\ the\ most\ similar\ clusters\ and\ merge$

Phase 3: Coverage Enhancement

10. $while\ W\ has\ words\ yet\ to\ be\ considered$
11. $\quad w = \arg\max_{w \in W} |D_w - \cup_{c \in C} c|$
12. $\quad if\ D_w\ overlaps\ with\ only\ one\ c \in C\ and\ also\ has\ max\ similarity\ with\ it$
13. $\quad\quad merge\ D_w\ with\ that\ cluster$
14. $Output\ C\ \&\ R$

are exactly k clusters left. Unlike other algorithms, RGC-D does not guarantee that all documents would be clustered. Many documents may remain unassigned to any cluster at the end of the second phase. The third phase tries to bring as many unclustered documents into the existing k clusters as possible.

Rules from RGC-D are in the form of disjunctions of words. Thus, a rule $w_1 \vee w_2$ represents a cluster that contains those and only those documents from $\mathcal{D}$ that contain w_1 or w_2 or both. We represent the set of documents from $\mathcal{D}$ that contain a word w by D_w. Further, for computing similarities between clusters, we use the cosine similarity between the respective centroid vectors. We describe each of the three RGC-D phases in detail below:

- **Cluster Generation:** We take all the words among documents in the input corpus, $\mathcal{D}$, and rank them using Centroid-Similarity Ranking (Balachandran *et al.*, 2009) as shown in line 1 of Algorithm 3. The top-t words from the ranking are used as candidates to create clusters in the first phase. Then, we take the word that covers most documents in $\mathcal{D}$ (line 2) and create the first cluster D_w which has

an associated rule w. For any w' that is considered thereafter, it checks whether the associated $D_{w'}$ has overlaps with any existing clusters; if not, a new cluster is created (line 4-5). If it so happens that $D_{w'}$ has overlaps with some existing clusters, we take the subset of clusters with which $D_{w'}$ overlaps, and also has a high similarity (using a threshold γ), and merge all of them together with $D_{w'}$ to form a new larger cluster (in lieu of the smaller ones in C which were involved in the merging). The merging process forms lines 6-7.

- **Cluster Merging:** At the end of the cluster generation phase, we may be left with more clusters than k; this is so since t usually needs to be chosen to be much higher than k to account for the mergers. We simply merge the most similar pair of clusters as in Hierarchical Agglomerative Clustering (Jain *et al.*, 1999) as shown in lines 8-9, until we end up with exactly k clusters.

- **Coverage Enhancement:** The k clusters after merging may not cover all the documents, since there could be documents in $\mathcal{D}$ that do not contain any word from W_α. In the coverage enhancement phase, we attempt to bring such documents into one of the k clusters. We do this by considering words in the vocabulary W that were not in W_α in the decreasing order of the number of new documents (documents not yet covered by the k clusters) that they cover. If a word w thus considered has a corresponding D_w that overlaps with multiple clusters among the k clusters, we discard it and move on to the next word. If the overlap is just with the one cluster with which D_w has maximum similarity, a merger is performed, and the associated cluster's rule is modified by adding w. Finally, the k clusters are output.

In addition to k, RGC-D has two parameters t and γ, of which the former is actually controlled using a parameter α. We found that there are large ranges of these parameters to which RGC-D is not sensitive. α, that determines t (t would be the number of words that ensure that α fraction of the dataset is covered), could be set to anywhere between 0.4 and 0.7, and γ could vary between 0.85 and 0.95 without any major changes in the clustering output. We will use such choices for RGC-D (specifically, 0.7 for α and 0.9 for γ) , thus leaving it with only one effective parameter k. As already

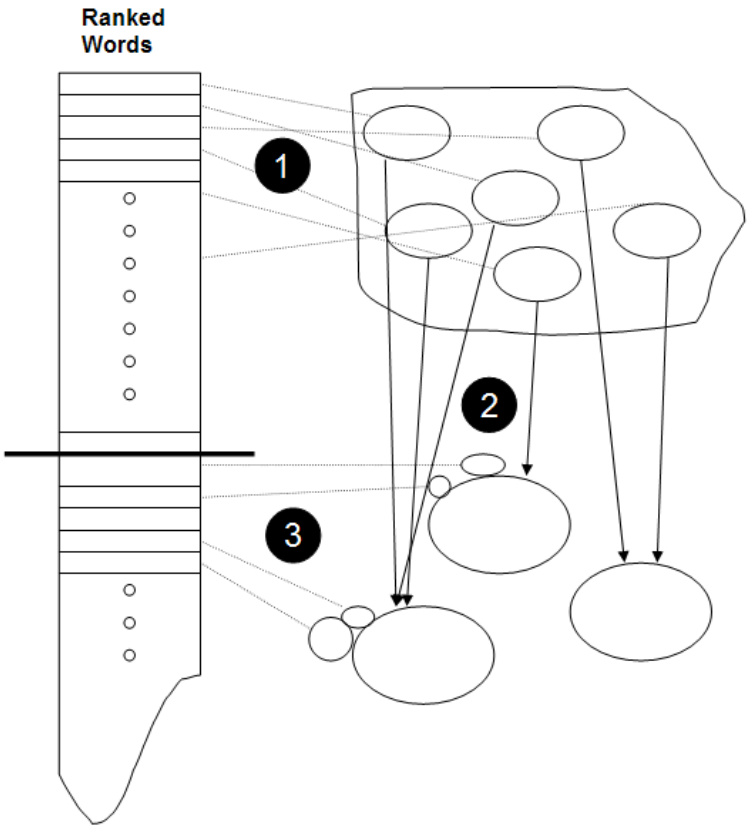

Figure 6.1: RGC-D Framework

Table 6.2: Datasets Used

Dataset	Documents	Words	#Classes
sports	8580	18324	7
k1b	2340	21839	6
ohscal3	2864	11465	3
r4	1013	7015	4
c3	3893	15490	3
cranmed	2431	41681	2

mentioned, unlike most other clustering methods, RGC-D does not guarantee that every document be assigned to one of the k clusters; the fraction of documents brought under the purview of the clustering, called *coverage*, is thus an indicator of the quality of the RGC-D clustering, since we would ideally like all documents to be clustered.

6.2.3 Experimental Evaluation

In this section, we will compare RGC-D against two techniques, the well-known k-means algorithm, and the interpretable clustering algorithm, UDT. RGC-D is a flat clus-

tering technique like k-means, but provides an added advantage that it provides clusters with interpretable descriptions; however, this comes with a cost, that of reduced accuracy of clustering when measures on traditional evaluation measures like purity. We focus on quantifying such loss of accuracy, in our comparison with k-means. In our comparison with UDT, we focus on the twin criteria of estimating how well RGC-D performs against UDT on both accuracy and conciseness of descriptions generated. We will describe the datasets and evaluation measures first, and then proceed to the results and discussion.

Datasets and Evaluation Measures:

Since our task is not a QA-specific task, we will use general text datasets from those described in Section 2.7.2. Statistics on the datasets we use are outlined in Table 6.2. The evaluation measure that concerns with interpretability is that of *rule length*. Though the rule length need not necessarily be directly related to the interpretability since small rules with complex words could be less interpretable than large rules with simple words, we will still use the rule length as a proxy of interpretability in our evaluation in the absence of better intuitive measures. Each RGC-D rule is a set of predicates each represented as single words; thus the length of the rule of a cluster is simply the number of words in the rule. The total rule length is then the sum of rule lengths across clusters.

We use F-measure as an evaluation measure to quantify the quality of clustering. Since not all documents in the corpus may be clustered by RGC-D unlike techniques like K-means, F-measure may be seen as a more suitable measure to evaluate clustering quality across RGC-D and K-Means. Since both RGC-D and K-means generate k clusters, we first establish a one-to-one correspondence between the generated clusters and the k classes from the labeled data (based on the class that has maximal representation in each cluster). Once such a correspondence is established, the cluster is treated as the retrieved set for the corresponding class, and the *precision* and *recall* measures are calculated using standard formulae. The F-measure (Powers, 2007) is then the harmonic mean of the precision and recall:

$$F - measure = 2 \times \frac{precision \times recall}{precision + recall}$$

For comparing RGC-D and UDT on clustering quality, we unfortunately do not have

the convenience of same number of clusters since UDT could generate far more leaf-level clusters than k. So, we resort to *net purity* for the comparative evaluation. Purity of an individual cluster in a clustering is determined as the fraction of the documents from the maximally represented class, in the cluster; thus, if a cluster comprising of 10 documents has 5 documents from one class (when the class labels of each data point is known as extrinsic information), and 3 and 2 documents from other classes, the purity would be $0.5(= 5/10)$. The *net purity* of a clustering is the cardinality weighted average of purities across clusters in it:

$$P(C) = \frac{\sum_{c \in C}(Purity(c) \times |c|)}{\sum_{c \in C} |c|}$$

Since not all documents may be covered by RGC-D and UDT clusterings, we penalize for low coverage by the following modification:

$$P(C) = \frac{\sum_{c \in C}(Purity(c) \times |c|)}{|\{d | d \notin C\}| + \sum_{c \in C} |c|}$$

Informally, all unclustered documents are assumed to be *misclustered*, in the above formulation of purity.

Issues in comparison with UDT: UDT does not have a k parameter, and thus could generate more than k clusters in the output. Though we can merge the clusters (by ORing the rules of the component clusters) until only k are left, the total rule length would remain constant when rulesets are merged using OR. Thus, for UDT, the total rule length would represent the total of the rule lengths across all the clusters it generates (the number of clusters could be significantly greater than k). Using net purity over all the clusters would however be advantageous to UDT since more clusters are likely to lead to better purity; at the extreme case where each object is in its own cluster, the purity would evaluate to 1.0. However, since we do not have a better platform for fairer comparison, we choose to use the purity measure over all clusters for UDT, despite such a measure being advantageous to it.

Evaluation on Total Rule Length:

We present the comparison between RGC-D and UDT on the *total rule length* in Table 6.3. As may be seen from the figure, those who would like to *assimilate* the

Table 6.3: Total Rule Length Evaluation

Dataset	Total Rule Length		% Reduction
	UDT	RGC-D	
sports	1381	1781	22.4%
k1b	3042	246	91.91%
ohscal3	2086	515	75.3%
r4	1652	216	86.9%
c3	1080	460	57.4%
cranmed	2037	231	88.6%

clustering by sifting through cluster descriptions of UDT would need to go through roughly 3-4 times as many words as for RGC-D. The rule length reductions achieved by RGC-D range from a low of 22% to a high of 92% and are around 70% on the average. Besides, the format of RGC-D rules are simpler by being just disjunctions of words, whereas UDT could have much more complex rules involving negations, frequency thresholds, conjunctions and disjunctions. Thus, the superiority of RGC-D over UDT in enabling easy understanding of the clustering is seen to be very stark.

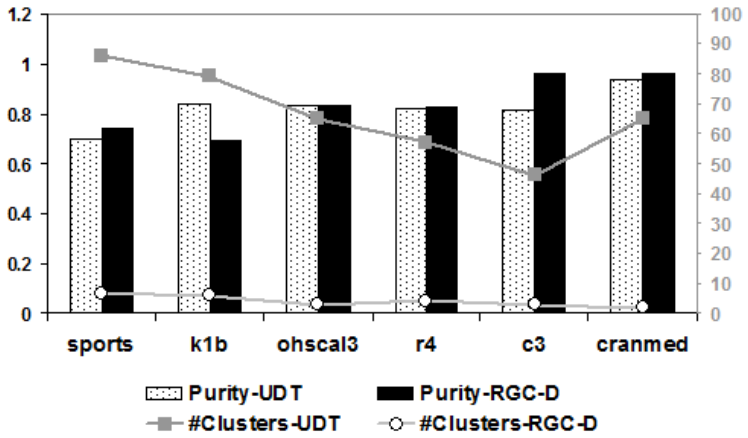

Figure 6.2: Purity Comparison with UDT (#Clusters on the Secondary Y-Axis)

Evaluation on Clustering Quality:

Since the number of clusters vary in UDT whereas it is same in the case of both RGC-D and k-means, we will illustrate the comparison of RGC-D against the two baseline techniques separately.

Figure 6.2, which plots the purity of the techniques, shows that UDT and RGC-D are fairly competitive. However, this needs to be read along with the number of clusters each of these techniques generate, that are plotted on the secondary (right) axis in the

same figure. The number of clusters generated by RGC-D are seen to be far fewer than those in UDT, with UDT generating as many as 7-8 times more clusters than RGC-D. Thus, despite the absolute purity numbers appearing to be competitive, it needs to be inferred that RGC-D generates far better results.

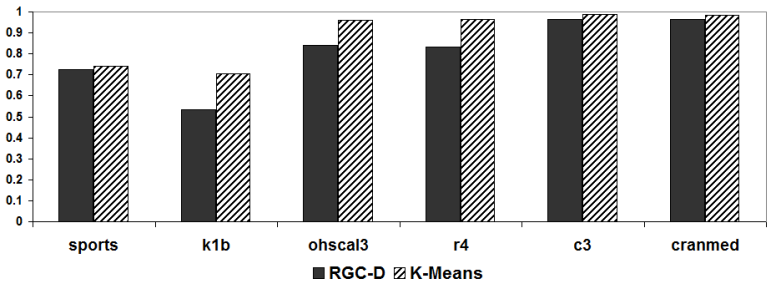

Figure 6.3: F-Measure Comparison with K-Means

Our comparison with k-means on F-measure is illustrated in Figure 6.3. The F-measure of RGC-D is expected to be lower than that of k-means; this may be seen as the cost of generating interpretable clusters (that the k-means does not generate). The measure of interest, however, is the amount of decrease in F-measure, and whether such decreases are tolerable in practical scenarios. The average F-measure of RGC-D clusters across datasets turn out to be 0.81, whereas the analogous measure for k-means is 0.89. Thus, the drop in purity is that of 8 percentage points; though significant, we argue that the added advantage of interpretability would offset it in scenarios where editing and reconfiguring clusters is an absolute necessity and manual inspection time is available only at a premium.

Sample RGC-D Rules:

Having evaluated RGC-D on quality and interpretability of the clustering, we now present some sample rules generated by RGC-D to give an intuitive feel of the understandability of the rules. For a few clusters from the reuters dataset, we provide some sample words that were included in the rules for each cluster. To recap, the rule $w_1 \vee w_2$ suggests that any document containing either of w_1 or w_2 would be part of the corresponding cluster. The sample rules are listed in Table 6.4. It may be intuitive to understand these rules, since they relate to matters of common knowledge (and are not very domain-specific) reuters dataset being a collection of newswire articles. The first rule is seen to have something to do with *coffee*, while the second has more to do with *oil* and *middle east*. The third one includes newsreports regarding international relations involving *russia*, *china* etc. The last rule may be inferred to belong to a cluster

Table 6.4: Sample Rules from RGC-D for Reuters Dataset

$grower \vee roaster \vee bag \vee crop \vee territory \vee bonus \vee \ldots$
$arab \vee arabian \vee alaska \vee pump \vee qatar \vee king \vee refinery \vee \ldots$
$moscow \vee chinese \vee veto \vee summit \vee korea \vee gatt \vee dispute \vee \ldots$
$westminster \vee mortgage \vee england \vee overdraft \vee gdp \vee merger \vee \ldots$

of news reports relating to the UK financial situation. If one needs to remove a specific concept, say *gatt* (which is the abbreviation for an international aggrement on tariffs and trade), one could simply delete the word from the third rule, and the underlying cluster is modified by simply removing all documents containing *gatt*.

6.2.4 Handling Multi-topic Datasets within the RGC-D Framework

Though partitional clustering where each document is assigned to a unique cluster is the most widely studied clustering setting, there could be many real-world scenarios where each document may be associated with multiple topics (making unique cluster assignment of documents inappropriate); this is specifically true in the case of clustering of document datasets that comprise widely varying and heterogeneous documents. For example, a news report describing an *Indian Premier League Cricket Match* could belong to various clusters such as *cricket*, *India* or *IPL* (abbreviation of the *Indian Premier League*) depending on whether there are enough other documents in the corpus that warrant a cluster for such topics. Some techniques from the fuzzy learning community (Alsberg, 1995) handle such cases by associating documents to multiple clusters with varying degrees.

We now briefly describe how we can handle clustering of multi-topic datasets within the framework illustrated in Figure 6.1. The first phase in RGC-D builds non-overlapping clusters; the second phase simply merges some of these clusters, thus retaining the disjointedness property. We propose that the cluster generation phase be amended by allowing for overlapping clusters in its output. The non-overlappingness is ensured in RGC-D by Lines 6-7 that induce a merger among all clusters with which D_w has a similarity exceeding a threshold γ, resulting in a single cluster that contains D_w (along with the other clusters used in the merger). We propose that these steps be replaced by a step where D_w be merged *separately* with each cluster with which it has a similarity of

at least γ. A similar amendment needs to be made in the coverage enhancement phase too where Lines 12-13 would be replaced by a step merging D_w with the cluster with which it has maximum similarity whether or not it overlaps with other clusters.

6.2.5 Closing Comments

In this section, we considered the problem of interpretable clustering to enable a user to easily filter out large collections of document datasets to identify (domain-specific) subsets of interest. We first outlined the motivation of the problem of interpretable and reconfigurable clustering in the scenario of quickly drilling down to a small subset of documents that could be of interest to enrich a domain-specific CBR system. We summarized related work of interest, and presented RGC-D, an interpretable clustering technique that employs a three phase methodology to arrive at a pre-specified number of clusters with associated simple rules that are formed by disjunctions of words. Our evaluation on clustering quality and interpretability against the state-of-the-art method of interpretable clustering, UDT, illustrates the superiority of RGC-D. We also show that the F-measure loss (with respect to k-means) to achieve interpretability is empirically seen to be around 8 percentage points on the average, across datasets. Further, we discussed how RGC-D can be extended to handle multi-topic datasets where each document needs to be assigned to multiple clusters.

Future work towards enhancing interpretability of clustering results could address issues like reducing the length of the rules. Though reducing the length of the rules would most likely reduce the quality as well, a framework where the user can specify the desired rule-length may be of interest in scenarios where the amount of time available for manual assimilation of clusters is known beforehand. This allows the algorithm to know the desired level of interpretability-accuracy trade-off, and heed to it during the cluster generation process. A potential area of future work is that of using tag-cloud like emphasizing of words in a rule (such as using different font sizes), so as to signify the estimated importance of various words in the rule. Another means of differential emphasis that directly falls out of the interpretable clustering framework is to assign an importance measure to each keyword that is directly related to the fraction of documents that gained membership in the cluster by virtue of the particular keyword. However, quantifying the incremental utility of such representations in aiding assimila-

tion of cluster content would be best done using carefully controlled user studies, which is outside the scope of the current work.

6.3 Similarity Measures for Microblog Data

We now turn our attention to the second problem outlined in Section 6.1, that of estimating similarity between microblog data. Accurate similarity assessment is among the major pre-requisites to bring in microblog data (or any kind of social media data) under the purview of Case-based Reasoning systems, since similarity assessment is one of the basic building blocks of any CBR system. In this section, we focus on similarity assessment between microblog data, especially tweets.

Tweets are basically short text messages authored by users of the social media platform called Twitter[2]; the length of each tweet is limited to 140 characters. Twitter has evolved to be a medium for sharing real-time and short status updates, unlike traditional social media where updates spanning multiple sentences and paragraphs is the norm. Tweets, in addition to the text content, have many kinds of metadata, such as the *time of the tweet*, *the author* and *geographic information* (in geo-tagged tweets). Twitter users may be mentioned in a tweet by including their twitter handle with the '@' prefix. Figure 6.4 illustrates a sample tweet from a well-known celebrity. Users in Twitter can follow other users, wherein their status updates would find presence in the follower's *stream*. Such follows relationship implicitly creates a graph structure involving twitter users. The imposition of the limit of length of the text content in Twitter has led to many people using uncanny abbreviations characteristic of textese (Deepak and Subramaniam, 2012). The various common noisy abbreviations of the word *tomorrow* include *2moro*, *tomm*, *tomora* and *tomra*. The extremely limited text content (given the length limit) and the usage of textese makes similarity assessment between tweets a challenging task. In this section, we explore methods to assess similarity between tweets exploiting both the text content and metadata in tweets.

[2]http://www.twitter.com accessed February 5^{th}, 2014

Figure 6.4: Sample Tweet

6.3.1 Related Work

The problem of finding similar microblog posts (e.g., tweets) is, to the best of our knowledge, a new problem that has not been addressed so far, in literature. In this section, we provide a brief overview of literature that is related to the problem. Firstly, we describe literature dealing with processing of microblog posts followed by work on processing SMS data (that are similar in lexical character to microblog posts). Lastly, we review prior work in the area of finding similarities between entities that utilize the graph structure and temporal information; these are pertinent in the context of the problem we address due to tweets having timestamps and being authored by people who have a social network presence.

Work on Microblog processing: Various retrieval-related tasks have been attempted on twitter and other microblog data. A recent work (Uysal and Croft, 2011) explores supervised learning approaches to identify tweets that are likely to be retweeted and proposes an approach for ranking users based on the likelihood of re-tweeting a particular tweet. Learning to Rank has been applied to identify features (Duan *et al.*, 2010) that could be used for general ranking of tweets to potentially replace the temporal recency based ranking that is employed on twitter and third-party twitter clients; the presence of a URL was found to be very useful feature therein. (Choudhury *et al.*, 2011) attempts to ensure diversity in the retrieval of tweets relevant to a particular topic (e.g., *Iran election*, *oil spill*); it describes a technique that solicits user input about the desired trade-off between diversity and homogeneity of content to be retrieved, and uses that to select a set of tweets in accordance to the specified diversity threshold. Sarma *et al.* (2010) evaluates mechanisms to gather user input for ranking in twitter-like forums and finds comparison-based ranking (where users are presented with two items and feedback on which is better is solicited) to be the most effective. Chen *et al.* (2010) develops a technique to recommend stories from extrinsic sources such as news articles and other content from the web, to twitter users; towards this, content relevance (assessed against the collection of tweets of the user, using traditional text similarity metrics) and social

voting are empirically identified as effective in choosing relevant content. *Buzzer* (Phelan *et al.*, 2009) identifies a set of articles from RSS feeds based on relevance to the top twitter trends identified based on recent public tweets (or a user's timeline, if provided). Qu and Liu (2011) propose using user tweet data to automatically group followers, based on seed users. SPOT (Perez *et al.*, 2011) is a technique to score twitter profiles based on suspicious activity. In addition to such retrieval and scoring tasks, twitter and other microblog data have been compared against traditional news media (Zhao *et al.*, 2011), analyzed wrt utility in enhancing informal communication at work (Zhao and Rosson, 2009) and used to identify patterns that could lead to a large follower base for individual users (Cha *et al.*, 2010).

Work on SMS Processing: SMS or text messages that are communicated over mobile devices also have lexical characteristics similar to that of microblog posts, due to length restrictions. However, it has been pointed out that SMS and twitter messages differ in the type of users, language (e.g., transliterated text being more common in SMS) and nature of topics discussed (Munro and Manning, 2012); thus, the utility of SMS processing techniques in processing Twitter data needs to be subjected to further verification. Of the major streams of work that deal with SMS data, *correcting* SMS data has been a subject of recent interest (Deepak and Subramaniam, 2012). While normalizing (i.e., correcting) microblog posts is a potential path towards enhancing retrieval on microblog data, we explore various means of enhancing similarity scoring directly on noisy text as an alternative approach towards the same goal of achieving noise-robust retrieval. Enabling effective querying over FAQ repositories using noisy SMS queries was the theme of a recent IR evaluation (Contractor *et al.*, 2011).

Work on Retrieving Entities: In our problem of finding similar tweets to a given tweet (we call this as the query tweet), one intuitive possibility is to prefer tweets from authors similar to that of the tweet in question; similarities between users in social networks would be useful for such consideration. Since content-wise similarity can be factored in directly by quantifying the similarity between the candidate tweets and the query tweet for our problem, we specifically focus on graph-based similarity approaches and exclude content based approaches (e.g., (Pennacchiotti and Gurumurthy, 2011; Diaz *et al.*, 2010)). Guy *et al.* (2010) assesses that familiarity based evidence (e.g., a user following another, direct communication between users) is less valuable than similarity-based evidence such as similar activity, similar tags etc. However, Han-

non *et al.* (2010) observes that the simple technique of suggesting the followers of a user as potential followees yields high accuracies. Another aspect of similarity is that pertaining to locations (Lee and Chung, 2011); geo-tagging however, not yet very popular in Twitter, unlike other networks such as FourSquare[3] that are centered on location-based search. Temporal information has been found to be useful for retrieval of time-stamped entities and has been incorporated successfully in classical techniques such as collaborative filtering (Ding *et al.*, 2006). For dynamic web sources such as RSS feeds, temporal recency is often the only criterion employed in ranking of items, and items are presented in the reverse chronological order. (Dong *et al.*, 2010) uses temporal activity of URLs in tweets to assess *freshness* of web pages in order to enhance traditional web search.

6.3.2 Problem Definition

Given a tweet q, a set of tweets T and k, we would like to identify an ordered set of k tweets from T, T_q, that are similar/relevant to q.

$$T_q = [t_1, t_2, \ldots, t_k]$$

Much like the problem addressed in Section 5.3, this boils down to designing a scoring function $S(Q, t)$ that estimates the similarity of each tweet t from T to Q. Given the scoring function $S(.,.)$, the i^{th} element of T_Q would then satisfy the following:

$$\forall t \in T - \{t_1, t_2, \ldots, t_{i-1}\}, S(q, t_i) \geq S(q, t)$$

This denotes that the scoring function scores t_i at least as much as every tweet not ahead of t_i in T. Informally, T_q contains the top-k tweets according to the $S(.,.)$ function in a non-increasing order of scores. We develop various scoring functions in the next section, and will evaluate them based on how well they model inter-tweet similarity in an empirical analysis. Based on results therein, we will drill-down on our recommendation of the best formulation for the $S(.,.)$ function.

[3]http://www.foursquare.com accessed February 5^{th}, 2014

6.3.3 Various Models for Scoring Tweets

We propose to exploit the following three kinds of resources for scoring tweets: (1) time of the tweet, (2) the author and the social network, (3) the textual content of the tweet. Each scoring function that we list herein uses a different way of exploiting one of the above resources.

Time-based Scoring (TS): The most intuitive way of using temporal information is probably that of considering the time difference between tweets. Tweets that are closer in time may be regarded as being likely to be more relevant to each other, in accordance with the expectation of the existence of a general temporal smoothness; infact, *reverse chronological ordering*, the standard ordering in most feedreaders uses such an intuition. This forms the basis of the first scoring function that we outline:

$$S_{TS}(q, t_i) = -1 \times |timestamp(t_i) - timestamp(q)|$$

Since we consistently use a higher value of the scoring function to indicate better relevance, we multiply the absolute difference in timestamps by -1 since time difference is inversely related to the temporal closeness of tweets.

Shared Connections (SC): We now seek to use author social network information induced by the *follows* relationship in Twitter. Since most twitter datasets that we have at our disposal do not include such *follows* relationship information and are just plain dumps of tweets, we induce a network by including an edge between two users if one of them has *sent* a tweet to the other. Thus,

$$A \leftrightarrow B \Rightarrow sentTweetTo(A, B) \vee sentTweetTo(B, A)$$

This defines an undirected (or bi-directed) graph where two users are connected if one of them has sent a tweet to the other.

The *Shared Connections* scoring function is based on the conjecture that a tweet is relevant to a user if she shares a lot of connections (i.e., twitter users) with the author of the tweet in question. We normalize the number of shared connections by the number of immediate connections across the two users. This leads to a scoring function like the

Jaccard Index[4], as below:

$$S_{SC}(q, t_i) = \frac{|\mathcal{N}(Author(q)) \cap \mathcal{N}(Author(t_i))|}{|\mathcal{N}(Author(q)) \cup \mathcal{N}(Author(t_i))|}$$

where $\mathcal{N}(A)$ denotes the set of immediate neighbors of user A. Among tweets authored by users that have no shared connections with the author of Q, a recency-based ordering is used (like in TS). This is inspired by a similar formulation that was proposed for predicting links between social network users Liben-Nowell and Kleinberg (2007).

Graph Distance (GD): SC conceptually scores two authors as having non-zero similarity iff they have at least one shared connection. However, it is obvious that interesting and relevant information could come from users with whom the author of Q does not even have one shared connection; in particular, some users may be only a few hops away from the tweet author and content from such users may be considered more relevant than those in a different connected component in the connections graph. GD formalizes a measure that calculates the similarity between tweets as inversely related to the distance between their authors on the network:

$$S_{GD}(q, t_i) = 1 - GraphDist(Author(q), Author(t_i))$$

$GraphDist(.,.)$ denotes the minimum number of hops that need to be traversed on the network graph (connections as defined earlier using $sentTweetTo(.,.)$ relations) to reach from one of the authors to the other. As in the case of SC, we use recency based ordering among tweets that are tied on the GD metric. The length of the shortest path between users was also previously employed for social network link prediction Liben-Nowell and Kleinberg (2007).

tf.idf Cosine (TC): Despite tweets having rich metadata like timestamps and author information, the main information in a tweet is contained within the text part of the tweet. TC uses the most common method of computing similarity over text data, the *tf-idf cosine* measure outlined in Section 2.5.3. We restate it here for completion:

[4]http://en.wikipedia.org/wiki/Jaccard_index accessed February 5^{th}, 2014

$$S_{TC}(q,t_i) = \frac{\sum_{w \in q \vee w \in t_i} tf\text{-}idf(w,q) \times tf\text{-}idf(w,t_i)}{\sqrt{\sum_{w \in q} tf\text{-}idf(w,q)^2} \times \sqrt{\sum_{w \in t_i} tf\text{-}idf(w,t_i)^2}}$$

where $tf\text{-}idf(w,q)$ denotes the value corresponding to the term w in the *tf-idf* vector of q (Refer Section 2.5.3).

Query-centric Similarity (QS): Consider the following tweets, q, t_1 and t_2:

$$q = \text{"}blasts\ in\ mumbai\ !!!\ :O\text{"}$$

$$t_1 = \text{"}mumbai\ blasts\ again\ omg\text{"}$$

$$t_2 = \text{"}mumbai\ blasts\ kill\ atleast\ ten\ people\text{"}$$

Let us, for simplicity, assume that the idf of all terms are 1.0, and that we do not consider exclamations and smileys in the similarity computation. Under such a model, $S_{TC}(.,.)$ scores t_1 at 0.577 wrt q whereas t_2 gets a lower score of 0.436. This is so since t_2 has more words, and leads to a larger denominator in the $tf\text{-}idf\ cosine$ similarity computation. However, besides being relevant to q, t_2 is seen to provide additional information, whereas t_1 is mostly redundant with respect to the q. Due to the possibility of the extra information contained in t_2 being actually useful, we would like to score t_2 at least as much as t_1. Thus, we propose to modify the *tf-idf cosine* similarity measure to not penalize for extra information in t_i, thus, leading to the following formula:

$$S_{QS}(q,t_i) = \sum_{w \in q} (f(w,t_i) \times idf(w))$$

where $f(w,t_i)$ denotes the frequency of w in the tweet t_i. Under this scoring function, it is easy to note that both t_1 and t_2 are scored equally at 2.0. By using a query-centric formulation that sums over the query words, we ensure that t_i is not penalized for the extra information it contains.

Edit-distance Based Similarity (ED): The *140*-character restriction on twitter (and similar restrictions on other microblogging services) induce easy and real-time dissemination of content; however, very often this forces the twitter user to use obscure abbreviations for the sake of brevity. For example, *parliament* is often abbreviated to *parlmnt*

or *prlmnt* whereas *atlntc* is used to refer to *atlantic*. Such shortening of words causes similarity under-estimation in similarity measures that rely on occurrences of the same word in two tweets to quantify relevance. Levenstein distance (Levenshtein, 1966) is a popular technique that quantifies the distance between two strings as the number of character edits (i.e., additions, deletions and substitutions) required to transform one string to the other and has been found to be effective in entity extraction (Wang *et al.*, 2009). We outline a similarity measure between words that is based on Levenstein distance, and further use it to develop a scoring function:

$$sim(w_1, w_2) = \begin{cases} 0.0, & if \;\; ed(w_1, w_2) > min\{len(w_1), len(w_2)\} \\ 1 - \frac{ed(w_1,w_2)}{min\{len(w_1),len(w_2)\}}, & otherwise \end{cases}$$

where $ed(.,.)$ measures the edit distance between the components, $len(w)$ denotes the number of characters in w. $sim(w_1, w_2)$ then measures the similarity between two words as inversely related to the edit distance measured as a fraction of the length of the shorter word. We use the length of the shorter word as a cut-off, and set the similarity of any pair that has an edit distance higher than the length of the shorter word, to 0.0. Our scoring function is then outlined as below, by aggregating the edit distance based similarity between pairs of words, one from Q and the other from t_i:

$$S_{ED}(q, t_i) = \prod_{w \in q}(1.0 + \sum_{w' \in t_i} sim(w, w'))$$

The addition of 1.0 to the inner sum is used to ensure that one of the inner sums evaluating to zero does not lead to an overall zero score.

Word Co-occurrences (WC): Tweets being short text snippets, any estimate of lexical similarity between them is heavily influenced by the very few words that occur in both tweets. The lack of redundancy in tweets, as observed earlier, brings the sparsity problem into prominence. Edit distance, while being able to alleviate problems due to misspellings, is unable to uncover semantic relatedness between words. For example, similarity measures discussed above would all rank the similarity between the words *Christmas* and *Yule* (the Northern European name for Christmas) at zero. However, due

to the relatedness of these words, over a large corpus of tweets, these words are likely to occur together in the same tweet; an example tweet from our corpus reads *'Yule would be the perfect day to take a day off and do my Christmas baking'*. Such co-occurrences were first explored for creating concept hierarchies from text data (Sanderson and Croft, 1999). The conditional probability of occurrence of a word w' given a word w is:

$$p(w'|w) = \frac{|\{d|d \in \mathcal{D} \wedge w \in d \wedge w' \in d\}|}{|\{d|d \in \mathcal{D} \wedge w \in d\}|}$$

where $\mathcal{D}$ denotes any large corpus of tweets (need not necessarily be T). We exploit such co-occurrences by using such conditional probabilities as a similarity measure between words, to define a scoring function as follows:

$$S_{WC}(q, t_i) = \prod_{w \in q}(1.0 + \sum_{w' \in t_i} p(w'|w) \times idf(w'))$$

We weigh similarities with high $idf(.)$ words higher, much like in Section 2.5.3.

Reply Correlations (RC): Another way of estimating co-occurrence based semantic relatedness of words is to exploit reply information. Its often the case that users respond to their followees' tweets; such replies can be treated as relevant to the followee's tweet, and hence could be treated as labeled data for retrieval of relevant tweets. However, in twitter and other microblogging sites, replies are not tagged with the tweet that is being replied to. We heuristically estimate that replies to the author within two hours of posting a tweet are replies to the tweet. An example tweet-reply pair thus extracted is given below:

Tweet	Im actually really excited for Friday what gunna happen to the government
Reply	I just hope there is no re-election

Given a set of such pairs $[t, r]$ denoted as $\mathcal{TR}$ that are extracted from a large corpus of tweets $\mathcal{D}$, we estimate a conditional probability between words in tweets and their replies, as follows:

$$p'(w'|w) = \frac{|\{[t, r]|[t, r] \in \mathcal{TR} \vee w \in t \vee w' \in r\}|}{|\{[t, r]|[t, r] \in \mathcal{TR} \vee w \in t\}|}$$

$p'(w'|w)$ estimates the probability of w' occurring in the reply given that w occurs in the tweet. Such word-correlations across question-answer pairs were implicitly used in (Xue *et al.*, 2008) leveraging translation models for improving retrieval accuracy in question answer forums. We now use a formulation similar to that in $S_{WC}(.,.)$ to develop a scoring function:

$$S_{RC}(q, t_i) = \prod_{w \in q}(1.0 + \sum_{w' \in t_i} p'(w'|w) \times idf(w'))$$

Wordnet Similarity (WS): Semantic relatedness between words may also be estimated using ontologies; the further apart two words are, in an ontology, the less likely they are, to be related. Wordnet[5] is a large ontology for the English language. Words in WordNet are organized into as many as $117k$ synonym sets (called *synsets*), each of which are associated with a brief textual definition called the *gloss*. Synsets associated with nouns are inter-related by relationships such as *is-a* and *part-of* whereas verb synsets are linked based on considerations such as one following another (e.g., *pay* typically follows *buy*). Among the various similarity measures proposed for quantifying pair-wise similarity between WordNet concepts (Pedersen *et al.*, 2004), we found the *Lesk* measure (Banerjee and Pedersen, 2002) to be most effective in estimating tweet relevance based on an empirical study. The *Lesk* measure estimates the similarity of a pair of words as being proportional to the extent of overlap of their dictionary definitions. We use the WordNet::Similarity package (Pedersen *et al.*, 2004) to compute word pair wise similarities. This leads to a scoring function as below:

$$\mathcal{S}_{ws}(\mathrm{II}, t_i) = \prod_{w \in \mathrm{II}}(1.0 + \sum_{w' \in t_i} lesk(w', w) \times idf(w'))$$

where $lesk(w', w)$ denotes the similarity between w' and w based on the *Lesk* similarity measure as applied on the WordNet ontology.

[5]http://wordnet.princeton.edu/ accessed February 5^{th}, 2014

6.3.4 Empirical Evaluation

We will now evaluate the different scoring functions outlined in the previous section to analyze their effectiveness in ranking relevant tweets highly. We will first start with describing the datasets, the experimental setup, and the evaluation measures used, and then present the results of the experiments.

Dataset and Experimental Setup:

Since the domain of our present study pertains to microblog data, we will use the only microblog dataset listed in Section 2.7. This dataset comprises of 977252 tweets authored by as many as 27165 twitter users. From these, we select 50 query tweets randomly to use as queries. For each query tweet q, we collect 200 most recent tweets from the dataset that were authored before q and share a common non-stopword term with q; this is used as T. Without the common non-stopword restriction, T would be left with mostly irrelevant tweets and would produce very few non-zero relevance judgements for the same labeling effort. We used 2-3 human labelers to judge the relevance of each of the $10k$ tweets (200 per query tweet, and 50 queries) to their respective query tweet. The total effort came to 25 hours, thus leading to roughly 9 seconds worth effort to judge the relevance of a single tweet. The corpus of 977052 tweets (i.e., the corpus excluding the 50 query tweets) was used as the dataset $\mathcal{D}$ to gather statistics on word and reply correlations for the $S_{WC}(.,.)$ and $S_{RC}(.,.)$ scoring functions.

Evaluation Measures:

We use the classical IR evaluation measures that were described in Section 5.5.2, viz., MRR, MAP, Precision and NDCG. Unless mentioned otherwise, we report the numbers for these measures on the top-10 results, for each query. Much like in Section 4.5.2, we use statistical significance assessments using randomization tests against a p-value of < 0.05.

Performance of the Time and Social Network-based Techniques:

We start by analyzing the results on the scoring functions that do not use the text content of the tweet; thus, we are left with TS, the temporal recency based scoring method, and the SC and GD rankings that use the social network information. We plot the various metrics in Figure 6.5. We also include the performance of a technique,

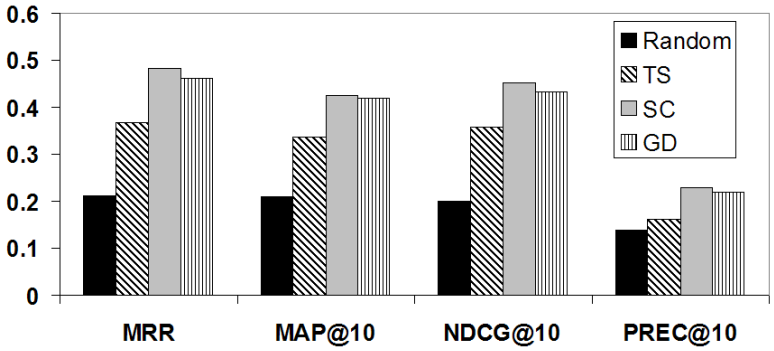

Figure 6.5: Time and Author based Techniques

Random, that retrieves tweets randomly (from T). It may be observed that the Time-based scoring (TS) is itself able to provide upto 75% gains in the MRR, MAP and NDCG measures over Random retrieval. The Social Network based measures (SC and GD) are seen to achieve upto 35% improvement on top of TS, across various metrics.

The performance of temporal recency based measure is an important indication of the relevance of time in the choice of topics in tweets; *the good performance of TS indicates that many tweets are likely to be about topics that have high temporal relevance*. We analyze the proportion of relevant tweets when only a subset of recent tweets from T are considered; this analysis is presented in Table 6.5. When only those tweets within 10 seconds of the query tweet are considered, it leads to an accuracy of as high as 30%; the accuracy gradually drops to 12.39%, the precision observed when all tweets in T are considered. This temporal correlation of relevance is not surprising since twitter is often used as a medium of real-time communication. It was further found that tweets in T were authored an average of 22.3 hours earlier than the corresponding query tweet, whereas the tweets in T that were marked relevant to the query were authored an average of 80 seconds earlier. This observation indicates that tweets that are similar to a query tweet are mostly found in the close *temporal neighborhood*; thus, a system that targets to find relevant tweets would almost necessarily have access to tweets in close temporal proximity to the query tweet. However, our technique could be easily adapted to do relevance based scoring on datasets that are temporally spaced out if some relevance information in the form of labeled tweets are available. The adaptation includes the introduction of prior weights for each scoring method that are tuned in cross-validation style using the available relevance information; the actual tweet scoring would then be determined by interpolating the prior weights with the query-specific

Time Difference	Precision in %
$<$ 10 secs	30.00%
$<$ 20 secs	31.11%
$<$ 40 secs	26.97%
$<$ 60 secs	23.45%
$<$ 80 secs	13.74%
All	12.39%

Table 6.5: Time Difference and Precision

Distance b/w Authors	Precision in %
1	66.67%
2	25.13%
3	14.27%
4	13.88%
5	13.74%
All	12.39%

Table 6.6: Social Network Distance and Precision

weights as determined within our composite technique.

That social network connections correlate well with relevance of tweets, as inferred from the good performance of the SC and GD techniques, indicates that people who are connected tend to have similar interests and tweet about similar topics around the same time. It is however, interesting that twitter users who have as many as 2-3 hops between them still exhibit some similarity in the topics of discussion. We analyze the average fraction of relevant tweets (i.e., Precision) when tweets from those at a particular distance from the author (in the social network induced using $sentTweetTo(.,.)$ relations) are considered, in Table 6.6. It is interesting to observe that upto 66.67% of tweets from immediate connections are found to be relevant; this indicates that immediate connections are highly likely to be tweeting about the same topic. Precision, as expected, steadily decreases with increasing social network distance, with only 25% of tweets from two hop distance being relevant. These have to be contrasted with an overall precision of 12.39% when tweets from all authors (whatever be the distance) are considered.

Performance of the Content based Techniques:

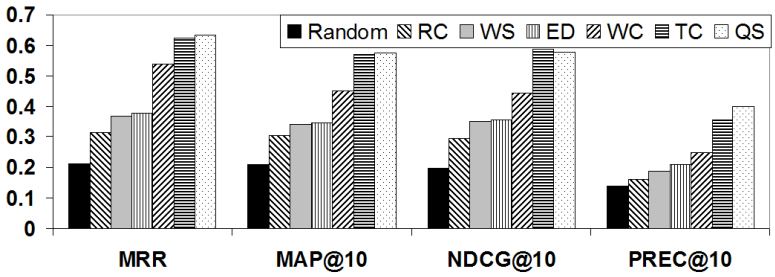

Figure 6.6: Textual Content based Techniques

We now analyze the performance of each of the content-based techniques in detail. Figure 6.6 plots the results on the content-based techniques on the various IR evaluation measures. We comment on the performance of each of these techniques, starting with the worst-performing one through the best one.

Reply Correlations (RC): RC was found to perform only around 50% better than Random Retrieval. We looked deeper into the kind of correlations RC was found to discover, in a bid to understand the poor performance. It was found that most replies were short and contained exclamatory remarks (e.g., *thats amazing*, *wow, its wonderful*, *lol*, *omg*), wishes (e.g., *congratulations*, *best wishes on your anniversary*) or short expressions (e.g., *thats good, thanks for sharing*, *thanks for that*). Thus, most of the replies do not have content very relevant to the tweet to which they are posed as replies. It was found that upto 30% of the replies had an exclamation mark, and roughly 10% contained the exclamatory word *omg*. This is not entirely unexpected, since microbloggers are bound by the character restriction and often reply compactly to tweets, and hence, replies more often than not, indicate just the polarity of the opinion.

WordNet Similarity (WS): WS performs around 75% better than Random Retrieval, and this illustrates that an ontology based similarity does improve retrieval performance. However, the performance improvement is not very substantial over RC. The top word pairs that were estimated to be similar according to the *Lesk* similarity metric were seen to have highly similar meanings, or different types of a same concept. The top few pairs included $[minute, second]$, $[north, south]$, $[weekend, week]$. It is easy to see that a tweet talking about *north* is unlikely to be relevant to be a tweet that has *south* within it. Despite such obvious shortcomings that limit the amount of improvements achieved by WS, ontology based similarity is seen to be useful to estimate relevance of many tweet pairs, thus achieving a significant gain over RC.

(w, w')	$p(w'\|w)$
(spears,britney)	0.0542
(degrees,number_token)	0.0493
(peanut,butter)	0.0481
(going,to)	0.0472
(minutes,number_token)	0.0435
(searching,for)	0.0431
(happy,!)	0.0429
(check,out)	0.0423
(anyone,?)	0.0414
(suggestions,?)	0.0389

Table 6.7: Top-10 Pairs According to WC

Edit Distance (ED): Similar to WS, The Edit Distance based approach performs roughly 75% better than random retrieval (and is hence, similar in performance to TS). We delved deeper into the words that were compared using edit distance, and regarded as highly similar. There were quite a few spurious matches, and as many as 9 spurious matches among the top-20 edit distance-wise similar word pairs. Such spurious pairs included $[protest, promise]$, $[breaking, being]$, $[against, again]$ and $[chocolate, coconut]$ while Edit Distance was able to rightly judge pairs such as $[thanksgiving, tgiving]$ and $[versions, version]$ as highly similar.

Word Correlations (WC): Word Correlations were found to be much more effective in judging tweet relevance, and score as much as 170% better than Random Retrieval, on an average. We list the top-10 correlated tokens (when ordered in the decreasing order of correlation) in Table 6.7. The exclamation mark, ! and the question mark, ? were treated as tokens whereas $number\_token$ is used to denote any number. The top-10 correlations seem quite interesting qualitatively, since the top few linked pairs do denote semantically related words; enhancing their similarity to something higher than 0.0 (which is what would be estimated under lexical similarity measures or even edit distance metrics) did help in improving retrieval accuracy substantially, as the results indicate.

tf-idf Cosine (TC): This measure, that is used as standard practice in various tasks, is seen to be very effective in judging similarity between tweets. Achieving as much as 3 times the performance of Random Retrieval over the various metrics, this shows that the limited redundancy in tweet data can be exploited effectively. The improvements

achieved by TC over RC and WS were found to be statistically significant at a p-value of < 0.05.

Query-Centric Similarity (QS): This technique, that uses a query-centric similarity measure, improves upon the tf-idf measure by using a query centric approach as described earlier. This is seen to outperform all techniques, including the TC method. It provides as much as 4 percentage point gains over TC on Precision, while outperforming TC by smaller margins on other metrics. QS, like TC, outperforms RC and WS statistically significantly; additionally, the improved performance of QS over ED was also found to be statistically significant.

6.3.5 Correlation Analysis and a Composite Technique

Having evaluated the various techniques in terms of their effectiveness in estimating relevance of tweets to a query tweet, we observe that content based techniques are most effective. This concurs well with the expectation since relevance is mostly assessed wrt the text content by the labelers, recency and author social network proximity being mostly not very apparent. Despite this, the metadata like time and author social network are likely to have some orthogonality with respect to content based relevance assessments. Time could be highly effective for ranking candidates for some queries; this is likely to be the case for extremely time-sensitive topics like real-time scores of an ongoing sporting contest. For social networks that are induced by tweet/reply information, there is a high likelihood of connections having common interests, and when such common interests peak (e.g., when the common interest is related to politics, activity would peak when there is an election), proximity in social network is likely to be a very accurate indicator of relevance. *Thus, a composite technique that is able to identify scenarios where specific techniques (e.g., content-based, time-based etc.) are likely to be more effective and weigh them highly for such cases, is likely to perform better than the separate methods.* In this section, we attempt to quantify the orthogonality between the various techniques, explore scoring of techniques, and develop a composite technique therein.

Correlation Analysis: We now analyze the orthogonality of the proposed techniques using the Pearson's co-efficient[6]. Given that we have 50 queries, we create a vector of length 50 whose i^{th} value denotes the precision ($@10$) obtained using that technique for the i^{th} query; a high value at the i^{th} index suggests that the technique performs very well for that query. We call such vectors as *precision vectors*. For a pair of techniques, we evaluate the Pearson Co-efficient between the precision vectors using the formula:

$$r = \frac{\sum_{i=1}^{50}(\mathcal{X}_i - \bar{\mathcal{X}})(\mathcal{Y}_i - \bar{\mathcal{Y}})}{\sqrt{\sum_{i=1}^{50}(\mathcal{X}_i - \bar{\mathcal{X}})^2} \times \sqrt{\sum_{i=1}^{50}(\mathcal{Y}_i - \bar{\mathcal{Y}})^2}}$$

where $\mathcal{X}$ and $\mathcal{Y}$ denote the precision vectors being compared and $\mathcal{X}_i$ stands for the i^{th} value of the vector $\mathcal{X}$. $\bar{\mathcal{X}}$ is a scalar denoting the mean of the values in the vector $\mathcal{X}$. The correlation co-efficient ranges between -1.0 and 1.0, the former denoting an inverse relation (e.g., high precision indexes of $\mathcal{X}$ corresponding to low precision in $\mathcal{Y}$) and the latter denoting a direct correlation, with a value of 0 denoting independence between the vectors. However, it has to be noted that correlation co-efficient only uncovers the existence or non-existence of a *linear relationship* and is unable to capture non-linear relationships; thus, inferences based on correlation co-efficients need to be taken with a pinch of salt.

Table 6.8 presents the correlation co-efficients among the three best performing content based techniques along with TS and SC (SC was seen to be the better among the social network based techniques). The table is obviously symmetric, with the entries in the diagonals (that correspond to comparing two identical precision vectors) assuming values of 1.0. We have highlighted the values that are at least 0.80, in the table. The recency and social network based technique are seen to be highly similar with a strong correlation of 0.85, whereas neither of them are highly correlated with any of the content based techniques. The content-based techniques (TC, QS and WC) seem to be highly correlated with each other with the exception of the (WC, QS) pair that scores at 0.68. With the values in the table, excluding the diagonal 1.0 entries, scoring at an average of 0.69 (which is still high enough as compared to independence, i.e., 0.0), all the techniques seem to have some high correlation with one another. However, that they are

[6]http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient accessed February 5^{th}, 2014

r	TS	SC	WC	TC	QS
TS	1.0	**0.85**	0.64	0.59	0.52
SC	**0.85**	1.0	0.67	0.68	0.61
WC	0.64	0.67	1.0	**0.80**	0.68
TC	0.59	0.68	**0.80**	1.0	**0.88**
QS	0.52	0.61	0.68	**0.88**	1.0

Table 6.8: Correlation Analysis

still rather far away from 1.0 on the average suggests that there is some orthogonality among the techniques that could be exploited. We will develop a technique to combine techniques in a bid to extract the best of the component techniques, in the next section.

Scoring Techniques based on an Estimate of Effectiveness: For a given scoring function S and a query q (and the associated candidate set of tweets, T), we would like to heuristically estimate the effectiveness of S in finding the top-k relevant tweets for q from T. This weighting score is intended to be an unsupervised one, retrieval being an unsupervised technique; thus, we will obviously not use any available relevance judgements for q in deriving an estimate of effectiveness for the $[S, q]$ pair. We score each tweet $t \in T$ wrt Q using the scoring function S and then normalize it to form a score S_N as below:

$$S_N(q,t) = \frac{S(q,t) - min\{S(q,t)|t \in T\}}{max\{S(q,t)|t \in T\} - min\{S(q,t)|t \in T\}}$$

$S_N(.,.)$ is always in the interval $(0, 1)$ for all scoring functions, but, the normalization process loses the information about the absolute values of $S(.,.)$ that were used to derive it. However, since the absolute values could vary widely among techniques (e.g., the scoring function for TS mostly has negative values whereas the scoring function for TC generates only positive scores), such a normalization helps the weighting score to be generic and technique independent.

Entropy: If the distribution of $S_N(.,.)$ were to be effective in assessing relevance, it should score the relevant tweets highly while scoring the irrelevant ones lowly. The ideal distribution would be that which scores all the relevant tweets at 1.0 and the irrelevant tweets at 0.0, leading to a low-entropy distribution. Under an unsupervised setting, due to being unable to correlate with relevance judgements, *we could simply*

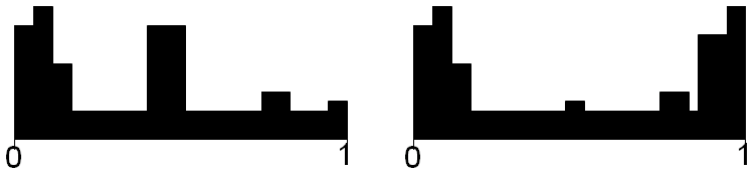

Figure 6.7: Distributions with Similar Entropy

prefer a scoring function that generates a low entropy distribution to a scoring function that induces a high entropy distribution.

Absolute Relevance: Consider Figure 6.7 which illustrates two distributions with probably similar entropy (Y-axis denotes the frequency of the value denoted by the corresponding X-value), due to both having two relatively significant peaks. Though entropy is unable to differentiate between the two, we would intuitively prefer the distribution on the right since there are more objects scored closer to 1.0. Ideally, we would like to provide k results, each of which according to S_N, are as close as possible to 1.0. Heuristically, we intend to *prefer those scoring functions whose S_N distributions score the top-k results as high as possible*. Due to normalization process, it is guaranteed that there would be one t_i at the high end, but, what we are interested in is in ensuring that there are at least k at the higher end.

Weighting Score: We use the above two considerations, (1) preference of low entropy, and (2) high scores for the top-k objects, to formulate a simple weighting function as follows:

$$w(S,q) = average(\text{top-}k(\{S_N(q,t)|t \in T\})) - entropy(\{S_N(q,t)|t \in T\})$$

where the $top\text{-}k(.)$ function takes a set of values and returns the $top\text{-}k$ values among them, and the $average(.)$ and $entropy(.)$ functions compute the average and entropy of the distributions respectively. Both these terms in the formulation are in the range $(0, 1)$ and the average of the $top\text{-}k$ terms (owing to it being the average of the highest k terms in a normalized distribution) is likely to be larger than the entropy of the distribution. This motivates the subtraction-based construction.

A Composite Technique: Having defined a weighting score for each combination

Evaluation Measure	Best Among Components	$\mathcal{S}_c$	% Impr. Recorded
MRR	0.633 (QS)	0.657	3.8%
MAP@10	0.575 (QS)	0.595	3.5%
NDCG@10	0.588 (TC)	0.594	1.0%
PREC@10	0.398 (QS)	0.400	0.5%

Table 6.9: Composite Technique Evaluation

of scoring function and query, we now outline an intuitive combined scoring function, given many scoring functions $\{S_1, S_2, \ldots, S_p\}$.

$$S_{\{S_1,\ldots,S_p\}}(q,t) = \sum_{i=1}^{p} \begin{cases} 0.0, \;\; if \;\; w(S_i,q) \leq 0.0 \\ \\ w(S_i,Q) \times S_{N_i}(q,t), otherwise \end{cases}$$

where $\mathcal{S}_{N_i}(.,.)$ denotes the normalized version of the $\mathcal{S}_i(.,.)$ function, computed as outlined earlier. For each query, the scoring functions are combined using a weighted sum construction; we use a cut-off of 0.0 thereby not allowing those (q, S_i) combinations that have a weighting score evaluating to negative to influence the scoring.

6.3.6 Empirical Results for the Composite Technique

We combine the top-5 content based techniques,WS, ED, WC, TC and QS with the recency based approach TS and the social network based approach SC to form a composite scoring function using the technique proposed in the previous section; we will refer to the combined technique as S_c hereafter. We now evaluate the composite technique, S_c wrt the component techniques on the various IR evaluation measures summarized in Section 6.3.4. Table 6.9 compares the performance of S_c against the best performing component technique on each of the measures. S_c is seen to outperform the components though only by very small margins. However, this establishes the utility of the weighting score formulation in leveraging the strengths of the various techniques to build a technique that outperforms the components, and establishes the combination as the preferred technique for retrieving similar/relevant tweets. The results on statistical significance are presented in Table 6.10. Though the improvements achieved by S_C

Technique Pair	Statistically ($p < 0.05$) Significant Metrics
QS over WC	*map, ndcg, prec*
QS over TC	*map, ndcg*
$\mathcal{S}_C$ over WC	*map, ndcg, prec*
$\mathcal{S}_C$ over TC	*map, ndcg*
$\mathcal{S}_C$ over QS	*map, ndcg*

Table 6.10: Statistical Significance

over QS and TC were found to be rather slim from Table 6.9, the improvements are seen to be statistically significant on the MAP and $NDCG$ measures. This establishes that S_C consistently outperforms them, despite the improvements being not very large. Table 6.10 only shows the statistical significance over a subset of technique pairs; S_C was seen to outperform all techniques other than TC and QS statistically significantly on all the four measures we considered.

Computational Complexity: *Computational Cost:* Each of our separate techniques take time of the order of $\mathcal{O}(|T|l^2p)$ where l denotes the number of tokens in a tweet, and p denotes the number of characters in a token (the edit distance calculations are linear in the number of characters in a token). We do not need to compute pair wise similarities between tokens at runtime (for $lesk(.,.)$ and $p(.|.)$) since they may be pre-computed for common pairs. Normalizing the scores to arrive at the S_N versions and final scoring are done serially and each take $\mathcal{O}(|T|)$, leading to an overall complexity of $\mathcal{O}(|\mathcal{T}|l^2p)$. l, the number of tokens in a tweet, is often 15-20 at max since tweets are limited to 140 characters. Thus, $\mathcal{S}_C$ is a very fast scoring technique for real scenarios.

6.3.7 Closing Comments

In this work, we analyzed the problem of finding similar/relevant microblog posts to a given microblog post. Though similarity search is a very popular technology in various other domains related to social media (such as finding similar user profiles, friend suggestions etc.), it has not been addressed in the specific case of microblog posts. Microblog posts in the popular microblogging service, Twitter, are short text snippets that are at most 140 characters in length. Associated with them are various kinds of metadata

such as a timestamp indicating the time of authorship, the twitter handle of the author and other less popular and optional metadata such as the geo-location of the tweet. We considered the utility of the timestamp and the author's social network, along with tweet content, in assessing the relevance of candidate tweets to a query tweet. Towards this, various intuitive techniques that separately exploit these kinds of information were developed. We analyzed them empirically and content based techniques were found to be most effective in ranking tweets. However, we observed through a correlation based analysis that there is significant orthogonality between the various techniques, and each of these are likely to be very useful in specific scenarios. We formulated a weighting score that heuristically estimates the effectiveness of specific techniques for given queries, in an unsupervised manner. We used such a weighting score to build a composite scoring function that assesses relevance using a linear combination of relevance assessments of the various component techniques, the relative weighting for each query being set according to the query-specific weighting scores estimated using our heuristics. An empirical evaluation illustrates that the composite technique improves upon the component techniques, many a time statistically significantly. Thus, the composite technique is seen to be the preferred technique for estimating tweet relevance, as seen from our empirical evaluation.

Since our dataset did not have many geo-coded tweets, the utility of geo-relevance in estimating tweet relevance has not been assessed. This could be an interesting direction for future work. Many recommendations and relevance assessments in new-age information systems are accompanied by an interpretable reason. Facebook[7] typically recommends connections with an accompanying line indicating the number of connections, whereas gmail[8] explains why it marked email threads as interesting by including an explanation indicating the reason; a common explanation is *"mainly because of your interactions with this thread"*. Similarly, interpretable relevance assessment could be useful in recommending relevant tweets too. Techniques to ensure diversity in retrieving relevant tweets is another natural next step to improve retrieval of relevant tweets. Our experiments show that the network-based similarity methods are less effective than the text similarity measures; in this context, exploring the usage of more advanced text processing methods (e.g., normalization methods to remove noise, language and topic

[7]http://www.facebook.com accessed February 5^{th}, 2014
[8]http://www.gmail.com accessed February 5^{th}, 2014

modeling methods) to improve similarity assesment between microblog data would be an interesting direction for future work.

6.4 Chapter Summary

In this chapter, we examined various ways in which data from unstructured text sources not in the format of problem-solution documents could be used to enhance Case-based Reasoning systems. We outlined various possibilities in using unstructured data ranging from discovering problem-solution documents from large unstructured text corpora procured from the web and/or the enterprise to presenting enhanced context to solutions retrieved by a CBR engine by exploiting related text documents. The conventional social media comprising of sources such as Facebook and Twitter could be used to enhance CBR utility by virtue of them containing time-relevant and/or more trusted information. We further outlined two problems, those of interpretable clustering and similarity assessment between microblog posts that serve to address some of the many possibilities of harnessing general unstructured text data. We presented RGC-D, an interpretable clustering algorithm that provides flat clustering of document datasets with associated word-based rules to serve as interpretable descriptions of each cluster. We illustrated through an extensive set of experiments that RGC-D does well in producing concise descriptions, and that the accuracy loss when compared to traditional clustering methods is limited to less than 10%. That may be seen as the cost of achieving the added advantage of interpretability, and may not be a large cost in many scenarios since interpretability could potentially save many man-hours in assimilating the clustering result. Secondly, we looked at the problem of scoring microblog posts with respect to a specified query microblog post, and outlined various intuitive scoring mechanisms that exploit one of the many features such as timestamp, author social network, and text content of the tweets. Based on an experimental analysis, it was seen that the content-based techniques fared the best. We then analyzed the orthogonality of the techniques using a classical correlation co-efficient, and devised a composite technique that combines a set of scoring methods into one composed method using a query-specific linear-combination based weighting. Our experiments with the composite technique establish the improved effectiveness of the composite technique in assessing tweet relevance. Through our novel proposals for interpretable clustering and microblog similarity as-

sessment, we have taken some small steps towards trying to incorporate unstructured data from diverse sources into Textual CBR systems.

CHAPTER 7

CONCLUSIONS

In this thesis, we explored many possibilities and challenges related to enhancing Textual CBR systems using text data of various kinds. We investigated the utility of concepts and techniques from knowledge management, especially, from fields such as text mining, natural language processing and information retrieval. Novel algorithms were developed to enhance Textual CBR systems in various ways such as extracting cases from text data, filtering cases to help maintain case bases, and enabling effective retrieval. This chapter concludes this thesis by summarizing the main contributions and listing down various promising directions for future work.

7.1 Contributions

We will now list the contributions of the thesis in the order of relevance to Textual CBR. Techniques that deal with textual problem-solution data are more central to the thesis due to their potential to influence Textual CBR more closely, whereas clustering and similarity assessment are less specific to Textual CBR. We list our contributions below:

1. **A technique for segmenting incident reports into cases:** In Chapter 4, we proposed *Correlation and Cohesion driven Segmentation (CCS)*, an algorithm that exploits statistical machine learning models to segment reports in an incident report collection into the component problem and solution parts. *CCS* assumes that the reports consistently contain the problem followed by the solution (and is applicable to any collection of reports where such a flow exists), wherein the problem is reduced to finding the partition point that splits the document putting the problem and solution parts at either side of the partition. *CCS* is applicable for a wide set of scenarios such as diagnosis reports and bug reports where the chronological ordering of the document leads to the problem-followed-by-solution format (i.e., the two-part format). Our technique uses language models

to learn the character of the problem and solution parts separately, and machine translation models to learn the correlation between the characters of these two segment types. Such models that are learnt on the corpus are used on each document to position the segmentation point at such a location in the document that maximizes the adherence of the document to the models; *CCS* uses an iterative EM formulation and refines the segmentation points with each iteration. We illustrated, through an extensive set of experiments on real-world data, that *CCS* outperforms the state-of-the-art segmentation algorithms by large margins in the segmentation of two-part text documents. It was also found that *CCS* is either insensitive to or degrades gracefully under various types of noise while continuing to outperform the baseline techniques even on significantly large amounts of noise.

2. **Compaction of Textual Case Bases:** Most problems from datasets derived from community-driven question answering systems (e.g., Yahoo! Answers, Quora etc.) have multiple solutions associated with a problem. We proposed a method of using standard statistical text similarity measures and solution popularity information to quantify the usability of a solution to a problem whose real solutions (i.e., solutions proposed for it) are available. We proposed fine grained evaluation measures that leverage the statistical usability estimates to evaluate the quality of a compacted case base. We illustrated that the compaction problem is non-trivial even with usability estimates due to various trade-offs involved. Whether generic solutions that work reasonably well for a wide variety of problems are to be retained in preference to specific solutions that are highly usable for a narrow range of problems is not immediately obvious, and there could be arguments that favor either extremes. Since such trade-offs are best judged by the user based on the usage scenario for the system, we model an intuitive user parameter that allows the user to specify the number of solutions desired for a problem, after case base compaction. We proposed a case base compaction technique, *GDO*, that can use such user preferences in compacting case bases using a greedy algorithm. Through extensive empirical analyses on various evaluation measures, we

establish the effectiveness of our technique in compacting textual case bases.

3. **A Query Suggestion technique for Textual Case Bases:** Textual CBR systems may not necessarily be successful in eliciting a full-blown description of the problem from the user since most users are conditioned by the extensive popularity of web search systems to provide short 2-3 word descriptions to a query system. Query suggestion is a technique employed by web search engines to provide user support for query authoring by providing real-time query suggestions as they type in queries on to the query text box. We proposed an improved query technique specialized to querying over textual case bases that exploits concepts from case based reasoning and language modeling. In particular, our technique prioritizes phrases relating to problems whose lexical neighborhood has well-aligned problems and solutions in the case base. Such a prioritization that is QA-aware is expected to avoid multiple steps of iterative query refinement, thus helping users to reach to the problem of interest rapidly. We illustrated, through extensive empirical analyses, that our technique outperforms the state-of-the-art method for generic query suggestions by large and statistically significant margins in the context of textual case bases.

4. **Interpretable Clustering Algorithm for Facilitating Drill-down to Domain-specific Text Documents:** Interpretable clustering is a useful tool for browsing document collections and choosing subsets of document collections of interest, to a specific task. The latter task has enormous significance in rapidly eliminating documents from a massive text dataset to find the small subset of documents relating to the domain of interest. We proposed an algorithm for flat interpretable clustering, *RGC-D*, where the cluster descriptions may be edited with the underlying cluster changing intuitively in accordance with the edit performed; this provides a useful tool to drill down even within clusters to select subsets of documents of interest. *RGC-D* provides cluster descriptions that are simply collections of words, where any document containing any of the words in the description would be part of the cluster. Our experimental study over many text document collections show that the loss of accuracy in order to achieve interpretability in *RGC-D* is limited

to reasonable bounds.

5. **Improved Similarity Estimation for Microblog Data:** To bring any form of data into a case base, an effective similarity measure for computing pair wise object similarity is a pre-requisite. Social media data differ from other text data in that it is often contaminated by various kinds of noise, the most prominent being that of conscious usage of unnatural abbreviations of words (e.g., *4m* for *from*). Such noise is most observed in length-limited media such as microblogs where the system enforces compactness, and users try to express more information under the length-constraint and end up using various kinds of abbreviations. Traditional text similarity measures such as tf-idf rely on occurrence of common words to quantify pair wise document similarity, and hence take a hit due to varying abbreviations for the same word across documents in microblogs. We investigated various kinds of similarity measures to provide accurate similarity assessments between microblog data, and found that a measure we propose, called *Query-centric Similarity* performs better than the state-of-the-art in estimating microblog similarity. Based on an analysis of orthogonality between similarity measures, we developed a method of combining various similarity measures into a composite method that outperforms each of the components in our empirical analysis.

7.2 Directions for Future Work

Our investigation into methods of enhancing Textual CBR systems by usage of general text data has yielded many techniques as described above. In this section, we highlight areas of shortcomings of the techniques we have proposed, and outline directions for further work under the theme of enhancing Textual CBR systems by attempting to incorporate the enormous amount of textual content generated each day.

1. **Extraction of Problem-Solution Segments from Text Documents:** One of the easiest ways to use text data in enriching Textual CBR systems is to directly inject more textual case bases into the case base, thus providing the system more data to

work with. This requires extraction of textual cases from text documents, as a first step. The *CCS* approach we proposed restricts itself to documents like incident reports authored in a two-part fashion with the problem followed by the solution. The general problem of extracting problem-solution segments from a text document that could contain arbitrary number of problem and solution segments strewn within it, would be a promising direction for future work. This would most likely require a supervised approach, where fingerprints for problems and solutions are learnt separately, and segments of text data are matched against them to adjudge them as either problem, or solution, or neither. The problems and solutions collected thus could be matched to form problem-solution pairs that would be used as cases.

2. **Stitching Text Fragments to form Problems and Solutions:** So far, we have considered how to reach to problems and solutions using a top-down approach, where collections of text documents are filtered to interesting documents, within which problem and solution segments are identified. In the era of social media and online forums, it could be the case that an initially posed problem is collaboratively refined, or a solution is arrived at through a series of posts. A simple scenario is the case of a forum thread that comprises of multiple posts, and the problem of summarizing the solution proposed using data across multiple posts. Such a bottom-up approach of finding statements that relate to a problem or a solution, and putting them together to form full problems and solutions would be of use to exploit such data for Textual CBR.

3. **Domain-specific tuning of Textual Similarity Measures:** Textual similarity measures such as tf-idf are generic; since CBR systems are usually domain-specific, we may like to introduce domain-specificity in the similarity measures used in Textual CBR systems. At the simplest level, a set of domain-specific terms may be learnt, and such terms may be given a higher weighting when computing cosine similarity between tf-idf vectors in the retrieval phase. Developing more sophisticated means of leveraging domain ontologies in similarity assessment would help enhance the utility of domain-specific Textual CBR systems.

4. **Text Reuse:** Despite some recent interest in text reuse techniques that help the user to effectively reuse the text content in the solutions of similar problems, text reuse still remains a challenging problem. Especially in newer domains such as social media, text reuse techniques may have to be blended with social network or other trust-indicating information to make automatic adaptation of historical textual content towards solving a new user problem.

5. **Usage of Extrinsic Knowledge Sources in Textual CBR:** There are a variety of public knowledge sources that are not in the form of problem-solution pairs; Wikipedia is one such general source of knowledge, whereas there are other domain-specific knowledge sources such as Wolfram Mathworld. Explorations into usage of such knowledge sources to enhance Textual CBR is still at an early stage (e.g., (Patelia *et al.*, 2011)) and is a potential direction for future research.

7.3 Thesis Summary

This thesis presented our work on addressing issues around knowledge management for text data with a focus on leveraging general text data to enhance Textual Case-based Reasoning systems. We illustrated the immense amounts of data that are available in various sources such as the web and the enterprise, and outlined various challenges in effectively utilizing them in knowledge reuse systems such as Case-based Reasoning.

CBR systems work with problem-solution datasets; we proposed techniques to extract problem-solution data from certain specific kinds of text data such as incident and diagnosis reports. Our contributions in maintenance of textual case bases focused on exploiting statistical text similarity measures for estimating usability of a solution (for a problem), and techniques for using such usability measures to filter case bases. We also investigated ways to assist users in authoring problems by recommending query suggestions that seek to help them to articulate problems better, and also sway them towards using vocabulary similar to that used in the case base. Further, we designed techniques that work with raw text data and enable the user to quickly drill-down towards text documents specific to a domain of interest; the interpretable clustering technique proposed towards that end is shown to outperform current techniques for interpretable cluster-

ing. One of the major challenges in incorporating social media data in case bases is that of developing effective similarity measures between social media content, since similarity-based retrieval is a basic operation for any CBR system. We proposed a similarity measure tailored to microblog text that is seen to improve the accuracy of similarity assessment by reasonable margins.

While empirically evaluating the techniques on real-world data, we also listed known limitations of our techniques. There are many possible extensions to our work, and challenges that have not yet been adequately addressed in order to make easy usage of text data in Textual CBR systems a reality.

LIST OF PAPERS BASED ON THESIS

1. Query Suggestions for Textual Problem Solution Repositories
   Deepak P, Sutanu Chakraborti, Deepak Khemani
   35^{th} European Conference on Information Retrieval Research (ECIR 2013), Moscow, Russia, March 2013. pp. 569-581

2. Two-part segmentation of text documents
   Deepak P, Karthik Visweswariah, Nirmalie Wiratunga, Sadiq Sani
   21^{st} ACM International Conference on Information and Knowledge Management (CIKM 2012), Maui, Hawaii, USA, October 2012. pp 793-802

3. Finding Relevant Tweets
   Deepak P, Sutanu Chakraborti
   13^{th} International Conference on Web-Age Information Management (WAIM 2012), Harbin, China, August 2012. pp. 228-240

4. Interpretable and reconfigurable clustering of document datasets by deriving word-based rules
   Vipin Balachandran, Deepak P, Deepak Khemani
   Knowledge and Information Systems (KAIS), 32(3), 2012. pp. 475-503

5. More or better: on trade-offs in compacting textual problem solution repositories
   Deepak P, Sutanu Chakraborti, Deepak Khemani
   20^{th} ACM International Conference on Information and Knowledge Management (CIKM 2011), Glasgow, Scotland, UK, October 2011. pp. 2321-2324

REFERENCES

1. **Aamodt, A.** and **E. Plaza** (1994). Case-based reasoning; foundational issues, methodological variations, and system approaches. *AI Communications*, **7**(1), 39–59.

2. **Aas, K.** and **L. Eikvil** (1999). Text categorisation: A survey. Technical report, Norwegian Computing Center.

3. **Adeyanju, I.** (2011). Case reuse in textual case-based reasoning. Technical report, Robert Gordon University, Aberdeen, Scotland.

4. **Adeyanju, I.**, **N. Wiratunga**, **R. Lothian**, **S. Sripada**, and **L. Lamontagne**, Case retrieval reuse net (cr2n): An architecture for reuse of textual solutions. *In International Conference on Case-based Reasoning (ICCBR)*. 2009.

5. **Adeyanju, I.**, **N. Wiratunga**, **J. A. Recio-García**, and **R. Lothian**, Learning to author text with textual cbr. *In European Conference on Artificial Intelligence (ECAI)*. 2010.

6. **Agichtein, E.**, **C. Castillo**, **D. Donato**, **A. Gionis**, and **G. Mishne**, Finding high-quality content in social media. *In International Conference on Web Search and Data Mining (WSDM)*. 2008.

7. **Aha, D. W.** and **D. Kibler**, Instance-based learning algorithms. *In Machine Learning*. 1991.

8. **Alsberg, B.** (1995). Fast, fuzzy c-means clustering of data sets with many features. *Journal of Comput. Chem.*, **16**(4), 414–421.

9. **Badra, F.**, **J. Cojan**, **A. Cordier**, **J. Lieber**, **T. Meilender**, **A. Mille**, **P. Molli**, **E. Nauer**, **A. Napoli**, **H. Skaf-molli**, and **Y. Toussaint**, Knowledge acquisition and discovery for the textual case-based cooking system wikitaaable. *In International Conference on Case-based Reasoning (ICCBR)*. 2009.

10. **Baeza-Yates, R.** and **B. Ribeiro-Neto**, *Modern Information Retrieval*. Addison Wesley, Harlow, 1999.

11. **Balachandran, V.** (2009). Interpretable and editable clustering of document datasets by deriving word-based rules. Technical report, Indian Institute of Technology Madras.

12. **Balachandran, V.**, **D. Padmanabhan**, and **D. Khemani**, Interpretable and reconfigurable clustering of document datasets by deriving word-based rules. *In International Conference on Information and Knowledge Management (CIKM)*. 2009.

13. **Banerjee, S.** and **T. Pedersen**, An adapted lesk algorithm for word sense disambiguation using wordnet. *In Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*. 2002.

14. **Basak, J.** and **R. Krishnapuram** (2005). Interpretable hierarchical clustering by constructing an unsupervised decision tree. *IEEE Trans. Knowl. Data Eng.*, **17**(1), 121–132.

15. **Bast, H.**, Type less, find more: fast autocompletion search with a succinct index. *In International ACM SIGIR Conference (SIGIR)*. 2006.

16. **Beeferman, D.**, **A. Berger**, and **J. Lafferty** (1999). Statistical models for text segmentation. *Mach. Learn.*, **34**(1-3), 177–210.

17. **Berger, A.**, **R. Caruana**, **D. Cohn**, **D. Freitag**, and **V. Mittal**, Bridging the lexical chasm: statistical approaches to answer-finding. *In International ACM SIGIR Conference (SIGIR)*. 2000.

18. **Bhatia, S.**, **D. Majumdar**, and **P. Mitra**, Query suggestions in the absence of query logs. *In International ACM SIGIR Conference (SIGIR)*. 2011.

19. **Blei, D. M.**, **A. Y. Ng**, and **M. I. Jordan** (2003). Latent dirichlet allocation. *The Journal of machine Learning research*, **3**, 993–1022.

20. **Boldi, P.**, **F. Bonchi**, **C. Castillo**, **D. Donato**, and **S. Vigna**, Query suggestions using query-flow graphs. *In Workshop on Web Search Click Data*. 2009.

21. **Boley, D.** (1998). Hierarchical taxonomies using divisive partitioning. Technical report, University of Minnesota.

22. **Breslow, L. A.** and **D. W. Aha** (1997). Simplifying decision trees: A survey. *Knowl. Eng. Rev.*, **12**(1), 1–40.

23. **Bridge, D.** and **P. Healy** (2012). The ghostwriter-2.0 case-based reasoning system for making content suggestions to the authors of product reviews. *Knowledge-Based Systems*, **29**, 93–103.

24. **Brin, S.** and **L. Page**, The anatomy of a large-scale hypertextual web search engine. *In World Wide Web Conference (WWW)*. 1998.

25. **Brown, M.**, **C. Fortsch**, and **D. Wissmann**, Combining ir and cbr for middle ground text retrieval problems. *In AAAI 1998 Workshop on Textual Case-based Reasoning*. 1998.

26. **Brown, P. F.**, **J. Cocke**, **S. A. D. Pietra**, **V. J. D. Pietra**, **F. Jelinek**, **J. D. Lafferty**, **R. L. Mercer**, and **P. S. Roossin** (1990). A statistical approach to machine translation. *Comput. Linguist.*, **16**(2), 79–85.

27. **Brown, P. F.**, **V. J. Pietra**, **S. A. D. Pietra**, and **R. L. Mercer** (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, **19**(2), 263–311.

28. **Cao, H.**, **D. Jiang**, **J. Pei**, **Q. He**, **Z. Liao**, **E. Chen**, and **H. Li**, Context-aware query suggestion by mining click-through and session data. *In ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 2008.

29. **Catherine, R.**, **R. Gangadharaiah**, **K. Visweswariah**, and **D. Raghu**, Semi-supervised answer extraction from discussion forums. *In International Joint Conference on Natural Language Processing (IJCNLP)*. 2013.

30. **Catherine, R.**, **A. Singh**, **R. Gangadharaiah**, **D. Raghu**, and **K. Visweswariah**, Does similarity matter? the case of answer extraction from technical discussion forums. *In International Conference on Computational Linguistics (COLING)*. 2012.

31. **Cha, M.**, **H. Haddadi**, **F. Benevenuto**, and **K. P. Gummadi**, Measuring user influence in twitter: The million follower fallacy. *In AAAI Conference on Weblogs and Social Media (ICWSM)*. 2010.

32. **Chakaravarthy, V. T.**, **H. Gupta**, **P. Roy**, and **M. K. Mohania**, Efficiently linking text documents with relevant structured information. *In International Conference on Very Large Databases (VLDB)*. 2006.

33. **Chakrabarti, S.**, **S. Roy**, and **M. V. Soundalgekar** (2003). Fast and accurate text classification via multiple linear discriminant projections pp. *Very Large Databases J.*, **12**(2), 170–185.

34. **Chen, J.**, **R. Nairn**, **L. Nelson**, **M. S. Bernstein**, and **E. H. Chi**, Short and tweet: experiments on recommending content from information streams. *In ACM CHI Conference on Human Factors in Computing Systems (CHI)*. 2010.

35. **Choudhury, M. D.**, **S. Counts**, and **M. Czerwinski**, Identifying relevant social media content: leveraging information diversity and user cognition. *In ACM Conference on Hypertext and Hypermedia (HyperText)*. 2011.

36. **Clarke, C. L.**, **M. Kolla**, **G. V. Cormack**, **O. Vechtomova**, **A. Ashkan**, **S. Büttcher**, and **I. MacKinnon**, Novelty and diversity in information retrieval evaluation. *In International ACM SIGIR Conference (SIGIR)*. ACM, 2008.

37. **Cohen, W. W.**, Fast effective rule induction. *In International Conference on Machine Learning (ICML)*. 1995.

38. **Cohen, W. W.** and **Y. Singer**, A simple, fast, and effective rule learner. *In AAAI Conference on Artificial Intelligence (AAAI)*. 1999.

39. **Contractor, D.**, **L. V. Subramaniam**, **P. Deepak**, and **A. Mittal**, Text retrieval using sms queries: Datasets and overview of fire 2011 track on sms-based faq retrieval. *In Forum for Information Retrieval Evaluation (FIRE)*. 2011.

40. **Cormen, T. H.**, **C. Stein**, **R. L. Rivest**, and **C. E. Leiserson**, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001, 2nd edition.

41. **Cucerzan, S.** and **R. W. White**, Query suggestion based on user landing pages. *In International ACM SIGIR Conference (SIGIR)*. 2007.

42. **Cutting, D. R.**, **D. R. Karger**, **J. O. Pedersen**, and **J. W. Tukey**, Scatter/gather: A cluster-based approach to browsing large document collections. *In International ACM SIGIR Conference (SIGIR)*. 1992.

43. **Deepak, P.** and **L. Subramaniam** (2012). Correcting sms text automatically. *CSI Communications*, **26**(2), 9–11.

44. **Dempster, A. P.**, **N. M. Laird**, and **D. B. Rubin** (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, **39**(1), 1–38.

45. **Diaz, F.**, **D. Metzler**, and **S. Amer-Yahia**, Relevance and ranking in online dating systems. *In International ACM SIGIR Conference (SIGIR)*. 2010.

46. **Ding, Y.**, **X. Li**, and **M. E. Orlowska**, Recency-based collaborative filtering. *In Australasian Database Conference (ADC)*. 2006.

47. **Dong, A.**, **R. Zhang**, **P. Kolari**, **J. Bai**, **F. Diaz**, **Y. Chang**, **Z. Zheng**, and **H. Zha**, Time is of the essence: improving recency ranking using twitter data. *In World Wide Web Conference (WWW)*. 2010.

48. **Duan, Y.**, **L. Jiang**, **T. Qin**, **M. Zhou**, and **H.-Y. Shum**, An empirical study on learning to rank of tweets. *In International Conference on Computational Linguistics (COLING)*. 2010.

49. **Fagin, R.**, **R. Kumar**, **K. S. McCurley**, **J. Novak**, **D. Sivakumar**, **J. A. Tomlin**, and **D. P. Williamson**, Searching the workplace web. *In World Wide Web Conference (WWW)*. 2003.

50. **Ferrucci, D. A.**, **E. W. Brown**, **J. Chu-Carroll**, **J. Fan**, **D. Gondek**, **A. Kalyanpur**, **A. Lally**, **J. W. Murdock**, **E. Nyberg**, **J. M. Prager**, **N. Schlaefer**, and **C. A. Welty** (2010). Building watson: An overview of the deepqa project. *AI Magazine*, **31**(3), 59–79.

51. **Feuer, A.**, **S. Savev**, and **J. A. Aslam**, Evaluation of phrasal query suggestions. *In International Conference on Information and Knowledge Management (CIKM)*. 2007.

52. **Fisher, D. H.** (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 139–172.

53. **Fonseca, B. M.**, **P. Golgher**, **B. Pôssas**, **B. Ribeiro-Neto**, and **N. Ziviani**, Concept-based interactive query expansion. *In International Conference on Information and Knowledge Management (CIKM)*. 2005.

54. **Gandhe, A.**, **D. Raghu**, and **R. Catherine**, Domain adaptive answer extraction for discussion boards. *In World Wife Web Conference (WWW)*. 2012.

55. **Gao, B.** and **Ester**, Cluster description formats, problems and algorithms. *In SIAM International Conference on Data Mining (SDM)*. 2006.

56. **Gray, R.**, **N. B. Ellison**, **J. Vitak**, and **C. Lampe**, Who wants to know?: question-asking and answering practices among facebook users. *In ACM International Conference on Computer Supported Cooperative Work (CSCW)*. 2013.

57. **Guy, I.**, **M. Jacovi**, **A. Perer**, **I. Ronen**, and **E. Uziel**, Same places, same things, same people?: mining user similarity on social media. *In ACM International Conference on Computer Supported Cooperative Work (CSCW)*. 2010.

58. **Halvey, M. J.** and **M. T. Keane**, An assessment of tag presentation techniques. *In World Wide Web Conference (WWW)*. 2007.

59. **Hannon, J.**, **M. Bennett**, and **B. Smyth**, Recommending twitter users to follow using content and collaborative filtering approaches. *In ACM Conference on Recommender Systems (RecSys)*. 2010.

60. **Harris, M. D.**, Building a large-scale commercial nlg system for an emr. *In International Natural Language Generation Conference (INLG)*. 2008.

61. **Hearst, M. A.**, Multi-paragraph segmentation of expository text. *In Annual Meeting of the Association for Computational Linguistics (ACL)*. 1994.

62. **Huang, A.**, Similarity measures for text document clustering. *In New Zealand Computer Science Research Student Conference (NZCSRSC)*. 2008.

63. **Jain, A.**, **P. Sarda**, and **J. R. Haritsa**, Providing diversity in k-nearest neighbor query results. *In Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. 2004.

64. **Jain, A. K.**, **M. N. Murty**, and **P. J. Flynn** (1999). Data clustering: A review. *ACM Comput. Surv.*, **31**(3), 264–323.

65. **Jantke, K. P.** and **V. Dotsch**, The necessity of user guidance in case-based knowledge acquisition. *In International Florida Artificial Intelligence Research Society Conference (FLAIRS)*. 1997.

66. **Jeon, J.**, **W. B. Croft**, and **J. H. Lee**, Finding similar questions in large question and answer archives. *In International Conference on Information and Knowledge Management (CIKM)*. 2005.

67. **Joachims, T.**, Text categorization with suport vector machines: Learning with many relevant features. *In European Conference on Machine Learning (ECML)*. 1998.

68. **Jones, R.**, **B. Rey**, **O. Madani**, and **W. Greiner**, Generating query substitutions. *In World Wide Web Conference (WWW)*. 2006.

69. **Kazantseva, A.** and **S. Szpakowicz**, Linear text segmentation using affinity propagation. *In International Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2011.

70. **Kelly, D.** and **J. Teevan** (2003). Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, **37**(2), 18–28.

71. **Kibler, D.** and **D. W. Aha**, Learning representative exemplars of concepts: an initial case study. *In International Workshop on Machine Learning*. 1987.

72. **Kleinberg, J. M.** (1999). Authoritative sources in a hyperlinked environment. *J. ACM*, **46**(5), 604–632.

73. **Kohavi, R.**, A study of cross-validation and bootstrap for accuracy estimation and model selection. *In International Joint Conference on Artificial Intelligence (IJCAI)*. 1995.

74. **Kolodner, J. L.** (1992). An introduction to case-based reasoning. *Artif. Intell. Rev.*, **6**(1), 3–34.

75. **Kozima, H.**, Text segmentation based on similarity between words. *In Annual Meeting of the Association for Computational Linguistics (ACL)*. 1993.

76. **Kummamuru, K.**, **D. Padmanabhan**, **S. Roy**, and **L. V. Subramaniam**, Unsupervised segmentation of conversational transcripts. *In SIAM International Conference on Data Mining (SDM)*. 2008.

77. **Kummamuru, K.**, **D. Padmanabhan**, **S. Roy**, and **L. V. Subramaniam** (2009). Unsupervised segmentation of conversational transcripts. *Statistical Analysis and Data Mining*, **2**(4), 231–245.

78. **Kwok, C.**, **O. Etzioni**, and **D. S. Weld** (2001). Scaling question answering to the web. *ACM Trans. Inf. Syst.*, **19**(3), 242–262.

79. **Lamontagne, L.** and **G. Lapalme**, Textual reuse for email response. *In Advances in Case-Based Reasoning*. Springer, 2004, 242–256.

80. **Lee, M.-J.** and **C.-W. Chung**, A user similarity calculation based on the location for social network services. *In International Conference on Database Systems for Advanced Applications (DASFAA)*. 2011.

81. **Lenz, M.**, Textual cbr and information retrieval - a comparison. *In In Proceedings 6th German Workshop on CBR*. 1998.

82. **Lenz, M.**, **A. Hübner**, and **M. Kunze**, Textual cbr. *In Case-Based Reasoning Technology*. 1998.

83. **Levenshtein, V. I.** (1966). Binary codes capable of correcting deletions, insertions, and reversals. Technical report, Soviet Physics Doklady.

84. **Liben-Nowell, D.** and **J. Kleinberg** (2007). The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, **58**(7), 1019–1031. ISSN 1532-2882. URL `http://dx.doi.org/10.1002/asi.v58:7`.

85. **MacQueen, J. B.**, Some methods for classification and analysis of multivariate observations. *In 5th Sym. of Maths, Statistics and Probability*. 1967.

86. **Manning, C. D.**, **P. Raghavan**, and **H. Schütze**, *An Introduction to Information Retrieval*. Cambridge University Press, 2008.

87. **Massie, S.**, **N. Wiratunga**, **S. Craw**, **A. Donati**, and **E. Vicari**, From anomaly reports to cases. *In International Conference on Case-based Reasoning (ICCBR)*. 2007.

88. **Michalski, R. S.** and **R. E. Stepp**, Learning from observation: Conceptual clustering. *In Machine Learning: An Artificial Intelligence Approach*. 1983.

89. **Morris, M. R.**, **J. Teevan**, and **K. Panovich**, What do people ask their social networks, and why?: a survey study of status message q&a behavior. *In Computer Human Interaction*. 2010.

90. **Munro, R.** and **C. D. Manning**, Short message communications: Users, topics, and in-language processing. *In Proceedings of the 2Nd ACM Symposium on Computing for Development*, ACM DEV '12. ACM, New York, NY, USA, 2012. ISBN 978-1-4503-1262-2. URL `http://doi.acm.org/10.1145/2160601.2160607`.

91. **Ng, A. Y.** and **M. I. Jordan**, On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *In International Conference on Neural Information Processing Systems (NIPS)*. 2001.

92. **Padmanabhan, D.**, **D. Rao**, and **D. Khemani**, Differential voting in case based spam filtering. *In Industrial Conference on Data Mining - Posters*. 2006.

93. **Pan, R.**, **Q. Yang**, and **S. J. Pan** (2007). Mining competent case bases for case-based reasoning. *Artif. Intell.*, **171**(16-17), 1039–1068.

94. **Panovich, K.**, **R. Miller**, and **D. R. Karger**, Tie strength in question & answer on social network sites. *In ACM International Conference on Computer Supported Cooperative Work (CSCW)*. 2012.

95. **Passonneau, R. J.** and **D. J. Litman** (1997). Discourse segmentation by human and automated means. *Comput. Linguist.*, **23**(1), 103–139.

96. **Patelia, A.**, **S. Chakraborti**, and **N. Wiratunga**, Selective integration of background knowledge in tcbr systems. *In International Conference on Case-based Reasoning (ICCBR)*. 2011.

97. **Paul, S. A.**, **L. Hong**, and **E. H. Chi**, Is twitter a good place for asking questions? a characterization study. *In AAAI Conference on Weblogs and Social Media (ICWSM)*. 2011.

98. **Pedersen, T.**, **S. Patwardhan**, and **J. Michelizzi**, Wordnet: : Similarity - measuring the relatedness of concepts. *In AAAI Conference on Artificial Intelligence (AAAI)*. 2004.

99. **Pennacchiotti, M.** and **S. Gurumurthy**, Investigating topic models for social media user recommendation. *In WWW (Companion Volume)*. 2011.

100. **Perez, C.**, **M. Lemercier**, **B. Birregah**, and **A. Corpel**, Spot 1.0: Scoring suspicious profiles on twitter. *In IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 2011.

101. **Pevzner, L.** and **M. A. Hearst** (2002). A critique and improvement of an evaluation metric for text segmentation. *Comput. Linguist.*, **28**(1), 19–36.

102. **Phelan, O.**, **K. McCarthy**, and **B. Smyth**, Using twitter to recommend real-time topical news. ACM Conference on Recommender Systems (RecSys). 2009.

103. **Plaza, E.**, Semantics and experience in the future web. *In European Conference on Case-based Reasoning (ECCBR)*. 2008.

104. **Powers, D. M. W.** (2007). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. Technical Report SIE-07-001, School of Informatics and Engineering, Flinders University, Adelaide, Australia.

105. **Proctor, J. M.**, **I. Waldstein**, and **R. Weber**, Identifying facts for tcbr. *In Textual Case-Based Reasoning Workshop*. 2005.

106. **Qu, Z.** and **Y. Liu**, Interactive group suggesting for twitter. *In Annual Meeting of the Association for Computational Linguistics (ACL)*. 2011.

107. **Quinlan, J. R.** (1986). Induction of decision trees. *Mach. Learn.*, **1**(1), 81–106.

108. **Raghunandan, M. A.**, **N. Wiratunga**, **S. Chakraborti**, **S. Massie**, and **D. Khemani**, Evaluation measures for tcbr systems. *In European Conference on Case-based Reasoning (ECCBR)*. 2008.

109. **Resnick, P.**, **K. Kuwabara**, **R. Zeckhauser**, and **E. Friedman** (2000). Reputation systems. *Commun. ACM*, **43**(12), 45–48.

110. **Reynar, J. C.**, An automatic method of finding topic boundaries. *In Annual Meeting of the Association for Computational Linguistics (ACL)*. 1994.

111. **Reynar, J. C.** (1998). Topic Segmentation: Algorithms and Applications. Technical Report IRCS-98-21, University of Pennsylvania Institute for Research in Cognitive Science.

112. **Robertson, S.** (2004). Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation*, **60**(5), 503–520.

113. **Robertson, S.** and **H. Zaragoza** (2007). On rank-based effectiveness measures and optimization. *Inf. Retr.*, **10**, 321–339.

114. **Sanderson, M.** and **W. B. Croft**, Deriving concept hierarchies from text. *In International ACM SIGIR Conference (SIGIR)*. 1999.

115. **Santos, R. L.**, **C. Macdonald**, and **I. Ounis** (2013). Learning to rank query suggestions for adhoc and diversity search. *Information Retrieval*, 1–23.

116. **Sarma, A. D.**, **A. D. Sarma**, **S. Gollapudi**, and **R. Panigrahy**, Ranking mechanisms in twitter-like forums. *In International Conference on Web Search and Data Mining (WSDM)*. 2010.

117. **Shokouhi, M.**, Learning to personalize query auto-completion. *In International ACM SIGIR Conference (SIGIR)*. 2013.

118. **Shokouhi, S. V.**, **A. Aamodt**, and **P. Skalle**, A semi-automatic method for case acquisition in cbr a study in oil well drilling. *In Artificial Intelligence and Applications (AIA)*. 2010.

119. **Smucker, M. D.**, **J. Allan**, and **B. Carterette**, A comparison of statistical significance tests for information retrieval evaluation. *In International Conference on Information and Knowledge Management (CIKM)*. 2007.

120. **Smyth, B.**, **P.-A. Champin**, **P. Briggs**, and **M. Coyle** (2009). The case-based experience web. *University College of Dublin*.

121. **Smyth, B.** and **P. Cunningham**, The utility problem analysed: A case-based reasoning perspective. *In European Workshop on Case-based Reasoning (EWCBR)*. 1996.

122. **Smyth, B.** and **M. T. Keane**, Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. *In International Joint Conference on Artificial Intelligence (IJCAI)*. 1995.

123. **Smyth, B.** and **E. McKenna**, Modelling the competence of case-bases. *In EWCBR*. 1998.

124. **Smyth, B.** and **E. McKenna**, Building compact competent case-bases. *In Case-Based Reasoning Research and Development*. 1999.

125. **Song, F.** and **W. B. Croft**, A general language model for information retrieval. *In International Conference on Information and Knowledge Management (CIKM)*. 1999.

126. **Spink, A.**, **B. Jansen**, **D. Wolfram**, and **T. Saracevic** (2002). From e-sex to e-commerce: Web search changes. *Computer*, **35**(3), 107–109.

127. **Spink, A. H.**, **D. Wolfram**, **B. J. Jansen**, and **T. Saracevic** (2001). Searching the web : the public and their queries. *Journal of the American Society for Information Science*, **52**(3), 226–234.

128. **Stamatatos, E.**, **N. Fakotakis**, and **G. Kokkinakis** (2000). Automatic text categorization in terms of genre and author. *Computational linguistics*, **26**(4), 471–495.

129. **Steinbach, M.**, **G. Karypis**, and **V. Kumar**, A comparison of document clustering techniques. *In KDD Workshop on Text Mining*. 2000.

130. **Stéphane, N.**, **R. Hector**, and **L. L. J. Marc** (2010). Effective retrieval and new indexing method for case based reasoning: Application in chemical process design. *Eng. Appl. Artif. Intell.*, **23**(6), 880–894.

131. **Teevan, J.**, **M. R. Morris**, and **K. Panovich**, Factors affecting response quantity, quality, and speed for questions asked via social network status messages. *In AAAI Conference on Weblogs and Social Media (ICWSM)*. 2011.

132. **Uysal, I.** and **W. B. Croft**, User oriented tweet ranking: a filtering approach to microblogs. *In International Conference on Information and Knowledge Management (CIKM)*. 2011.

133. **Wang, W.**, **C. Xiao**, **X. Lin**, and **C. Zhang**, Efficient approximate entity extraction with edit distance constraints. *In Annual ACM SIGMOD Conference (SIGMOD)*. 2009.

134. **Weiss, S. M.** and **N. Indurkhya**, Lightweight rule induction. *In International Conference on Machine Learning (ICML)*. 2000.

135. **Wu, H. C.**, **R. W. P. Luk**, **K.-F. Wong**, and **K.-L. Kwok** (2008). Interpreting tf-idf term weights as making relevance decisions. *ACM Trans. Inf. Syst.*, **26**(3).

136. **Xue, X.**, **J. Jeon**, and **W. B. Croft**, Retrieval models for question and answer archives. *In International ACM SIGIR Conference (SIGIR)*. 2008.

137. **Yang, Q.** and **J. Zhu** (2001). A case-addition policy for case-base maintenance. *Computational Intelligence*, **17**(1), 250–262.

138. **Zhai, C.** and **J. D. Lafferty**, A study of smoothing methods for language models applied to Ad Hoc information retrieval. *In International ACM SIGIR Conference (SIGIR)*. 2001.

139. **Zhao, D.** and **M. B. Rosson**, How and why people twitter: the role that micro-blogging plays in informal communication at work. *In ACM international conference on Supporting group work (GROUP)*. 2009.

140. **Zhao, W. X.**, **J. Jiang**, **J. Weng**, **J. He**, **E.-P. Lim**, **H. Yan**, and **X. Li**, Comparing twitter and traditional media using topic models. *In European Conference on Information Retrieval (ECIR)*. 2011.

141. **Zhao, Y.** and **G. Karypis**, Evaluation of hierarchical clustering algorithms for document datasets. *In International Conference on Information and Knowledge Management (CIKM)*. 2002.

142. **Zhou, G.**, **L. Cai**, **J. Zhao**, and **K. Liu**, Phrase-based translation model for question retrieval in community question answer archives. *In Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT)*. 2011*a*.

143. **Zhou, T. C.**, **C.-Y. Lin**, **I. King**, **M. R. Lyu**, **Y.-I. Song**, and **Y. Cao**, Learning to suggest questions in online forums. *In AAAI Conference on Artificial Intelligence (AAAI)*. 2011*b*.

144. **Zhu, J.** and **Q. Yang**, Remembering to add: competence-preserving case-addition policies for case-base maintenance. *In International Joint Conference on Artificial Intelligence (IJCAI)*. 1999.